Industry Grade Project -1

Divya Devi V

14-May-2023

—

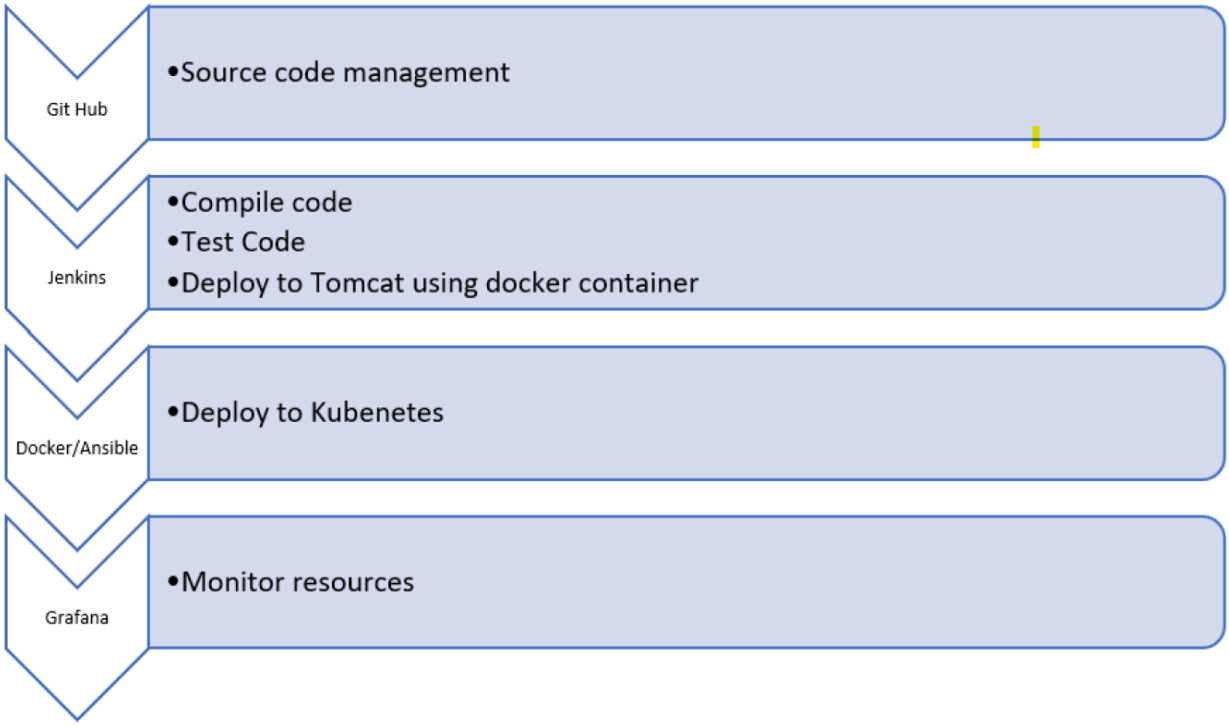
Devops Certification

—

Edureka

Building a CI-CD pipeline

The Process Flow to be followed to achieve this.



Explaining the Tasks that will be done.

- Push the code to our GitHub repository.
- Create a continuous integration pipeline using Jenkins to compile, test, and package the code
- Write Dockerfile to push the war file to the Tomcat server.
- Integrate Docker with Ansible and write the playbook.
- Deploy artifacts to the Kubernetes cluster
- Monitor resources using Grafana.

What Softwares will be installed?

1. Java
2. Maven
3. Git
4. Jenkins
5. Docker
6. Ansible
7. Kubernetes
8. Grafana
9. Prometheus

Resources Required for these

1. AWS Account
2. MOBA-Xterm/Putty
3. Github account
4. Docker-hub account

We will also add details on the same document on how these installations and resource setups were done.

Let us get started on the tasks one by one.

Task 1: Clone the project from the GitHub link shared in resources to your local machine. Build the code using Maven commands.

Also here am going to do the git and maven installation on AWS EC2 server.

Steps done to complete the task.

1. Creating a git hub account.
2. Clone the project from Edureka resources to your IDE.
3. Installing git bash client and push the code to your github account
4. Verify the status of the push.
5. Installing maven
6. Build the code with maven

What is GIT?

Git is free and open-source software for distributed version control, tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.

Creating a GIT hub account

- Go to https://github.com/join to create a GitHub account.
- You'll need to verify your email during the signup process.
- Select the free plan account and create one.

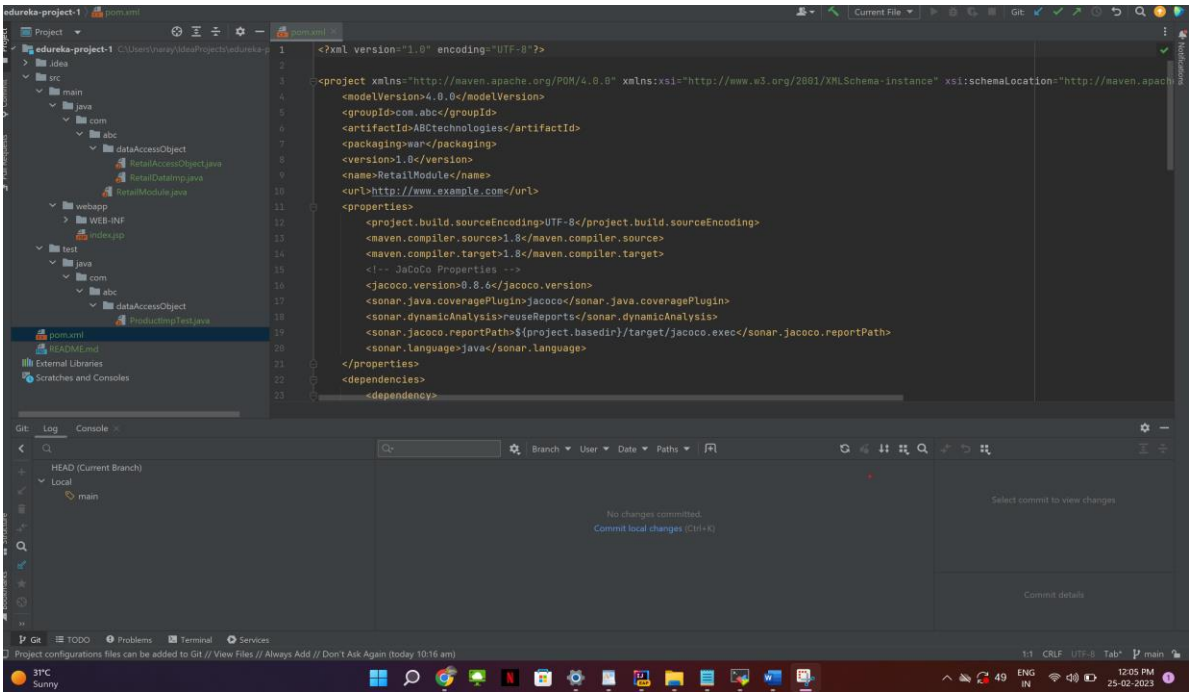
Successfully created github account.

Also created a initial repository with only a readme file.

https://github.com/divyadevivasudevan/edureka-project-1.git

Installed Git bash on my PC. Git BASH:: Git for Windows provides a BASH emulation used to run Git from the command line which is same as the linux env.

The git client is then used from our IDE -Intellij in my case. Copied the project from Edureka to my local

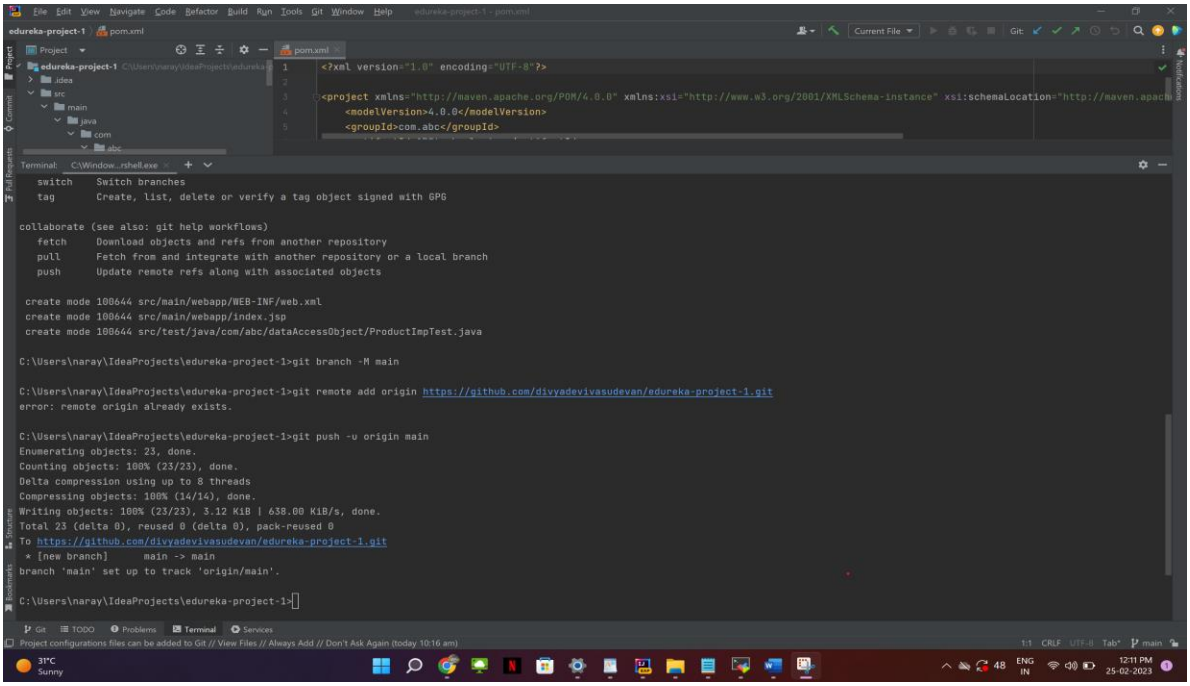


Committed and pushed the repo to my github remote repo using the git commands.
Git commands executed are:

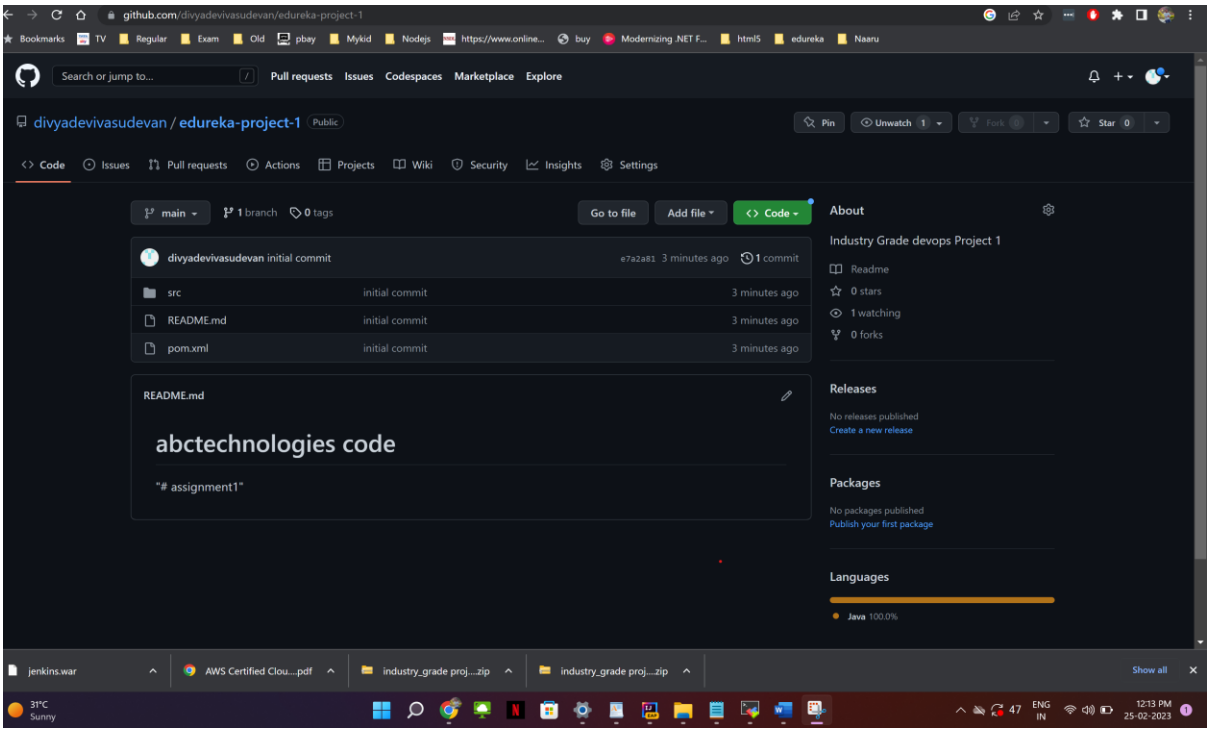
- git init for initializing a local repository
- git add . to add all your files that the local repository
- git commit -m 'commit message' to save the changes you made to those files

- git remote add origin https://github.com/divyadevivasudevan/edureka-project-1.git
- git branch -M main (creating a branch main)
- git push -u origin main – This pushes my code in the local to the main branch in the remote.

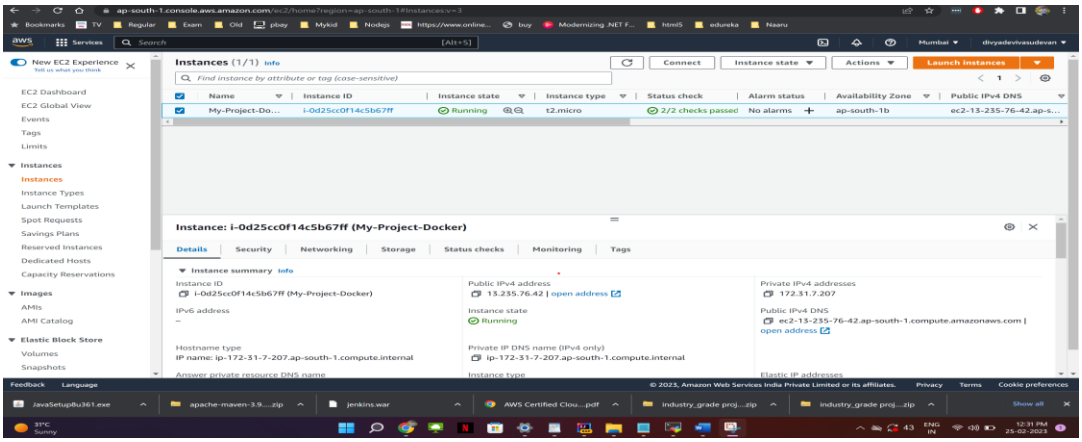
The Git logs can be seen..



Verified the Github whether commit and push was successful.

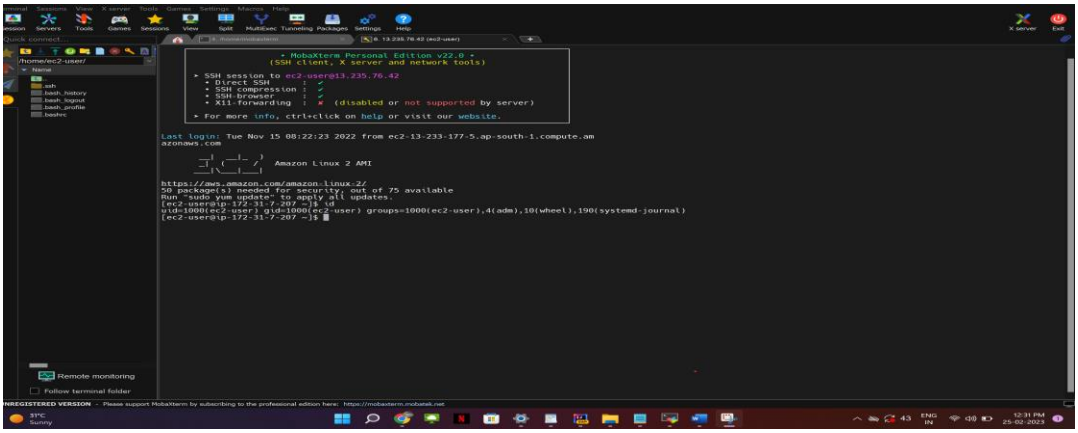


Launch an EC2 instance.

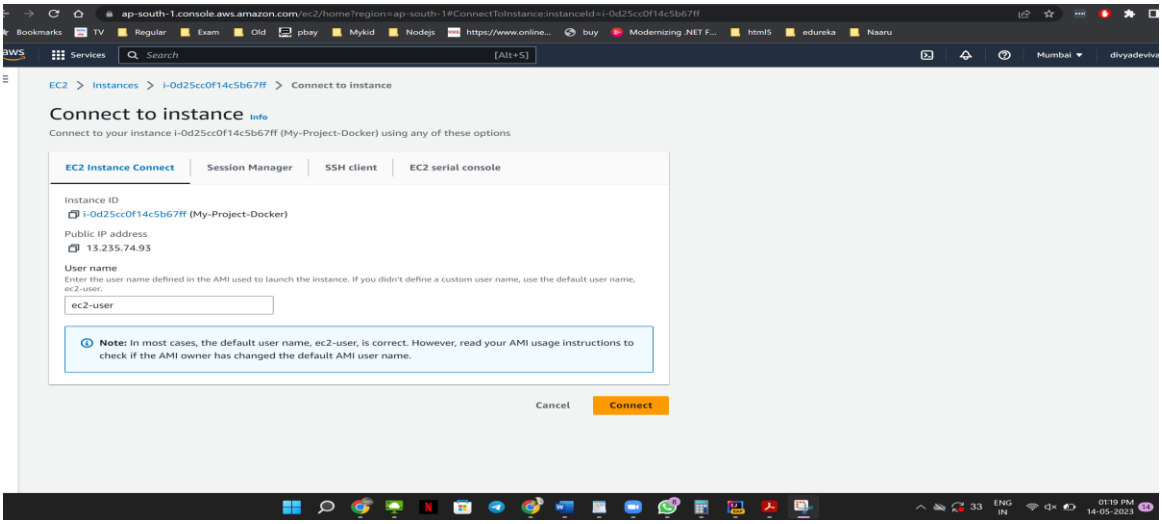


Connect to the instance with moba-xtrem or any ssh client using the PEM.

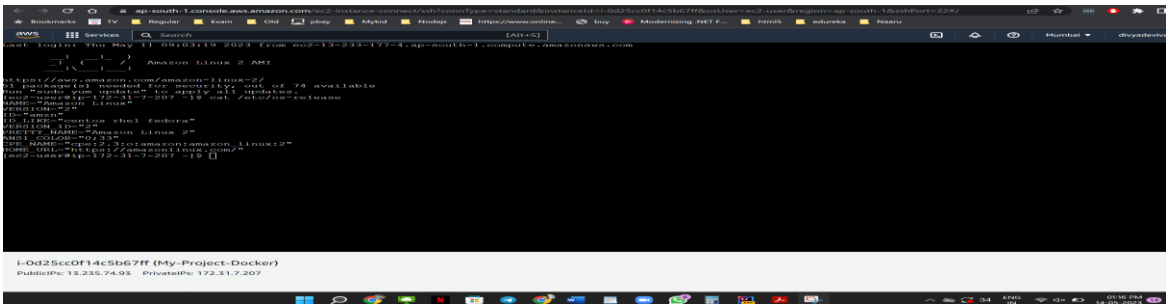
Moba-xtream:



Can also use the connect ssh client facility available in the EC2 portal which is much simpler in process.



Command to verify the OS version : cat /etc/os-release.

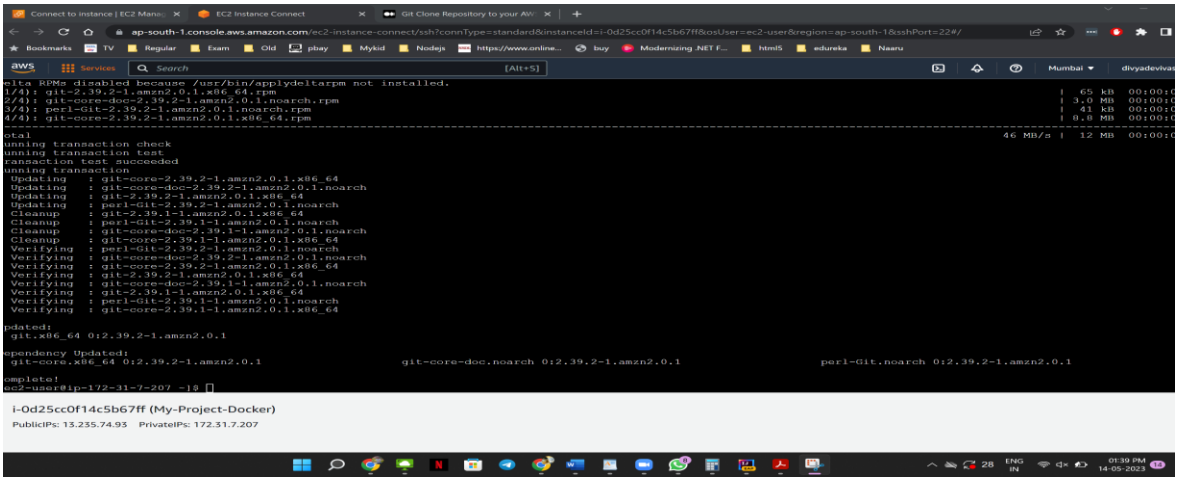


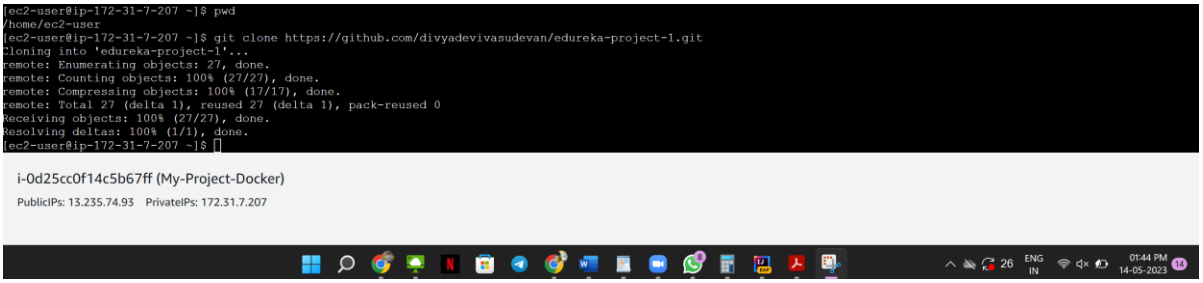
Installing GIT and MAVEN and Run MVN commands

Step 1:: Install Git -> Command : sudo yum install git -y

Step 2::clone the repo in ec2 ->Command – git clonehttps://github.com/divyadevivasudevan/edureka-project-1.git

Committed the installation script to git repo – file name git/installgit.sh





What is Maven?

Maven is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages.

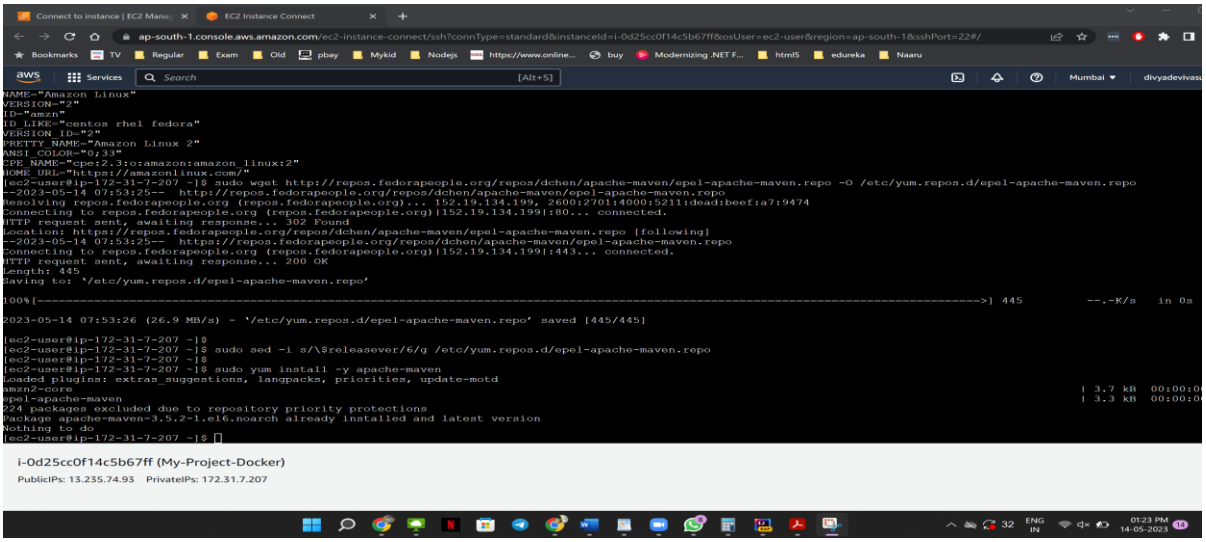
#Install maven in the server

sudo wget http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo

sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo

sudo yum install -y apache-maven

Committed the installation script to git repo – file name maven/installmaven.sh

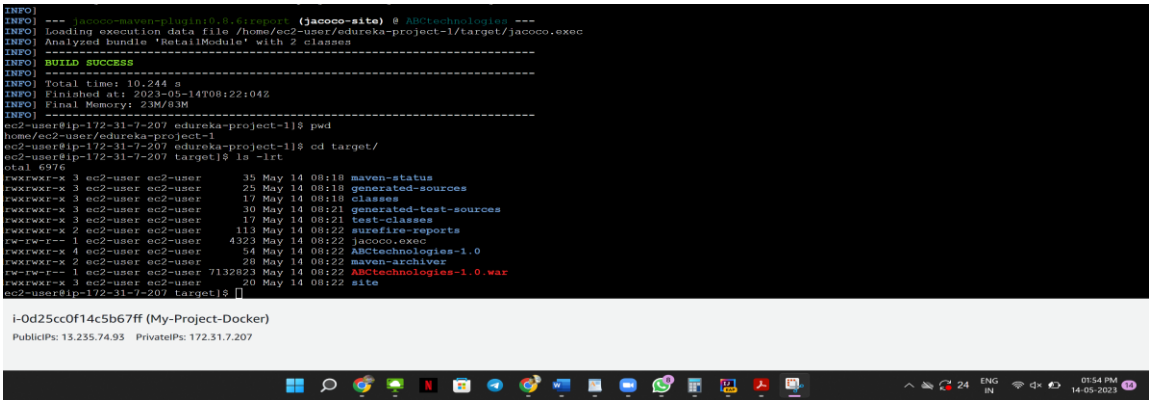


Basic Maven Commands :

1) mvn --version to find the version of the maven install

2) mvn compile – to compile the source code.
3) mvn test - to run the test of the project
4) mvn package – to build and package the artifacts of the project to jar or war

mvn package from ec2 :: added war plugin in the pom.xml and verified the /target directory



----------END of TASK 1----------

Task 2: Set up the Git repository and push the source code. Then, log in to Jenkins.

1. Create a build pipeline containing a job for each
   - One for compiling source code
   - Second for testing source code
   - Third for packing the code
2. Execute the CI/CD pipeline to execute the jobs created in step 1
3. Set up a master-slave node to distribute the tasks in the pipeline.

What is Jenkins?

Jenkins is an open-source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery.

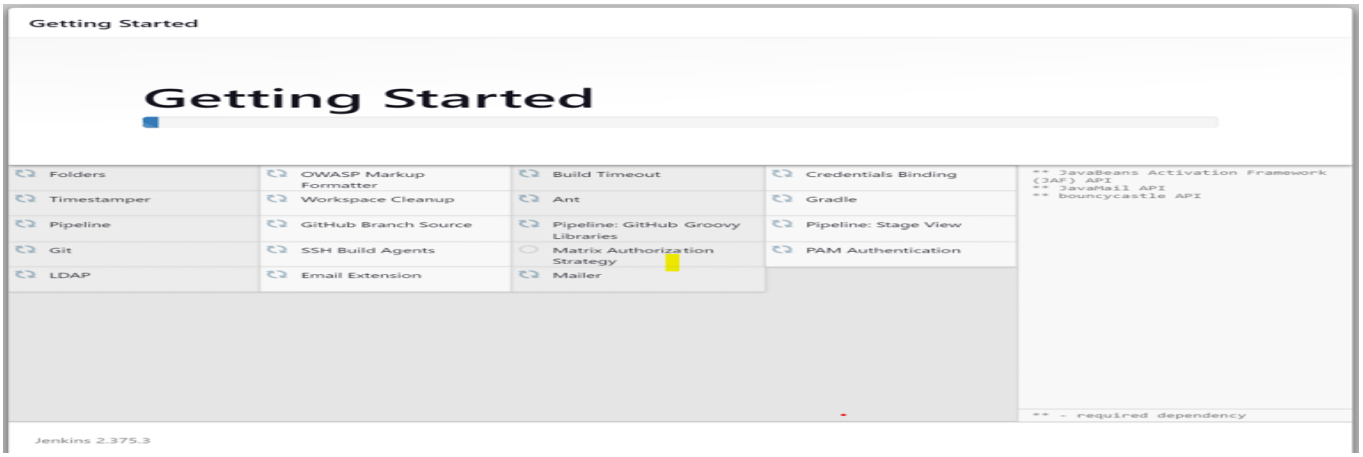
How to install Jenkins?

```
sudo -i

sudo yum update -y

#Install Jenkins :

sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-
stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
yum install jenkins
```

```
#Install epel Package:
amazon-linux-extras install epel
#Install Java:
amazon-linux-extras install java-openjdk11


#Start Jenkins:

# Start jenkins service
service jenkins start

# Setup Jenkins to start at boot,
chkconfig jenkins on

to get the admin password
cat /var/lib/jenkins/secrets/initialAdminPassword
```

Then the same can be accessed using the public IP : 8080 with admin password.

The screen will show a startup config, where you could choose to install the recommended plugins.

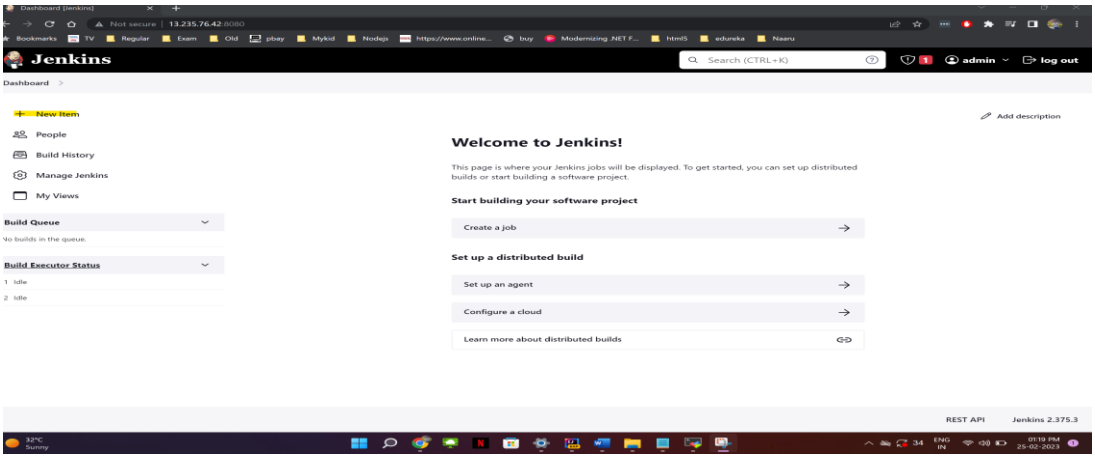


JOB Creation- Source Code compilation

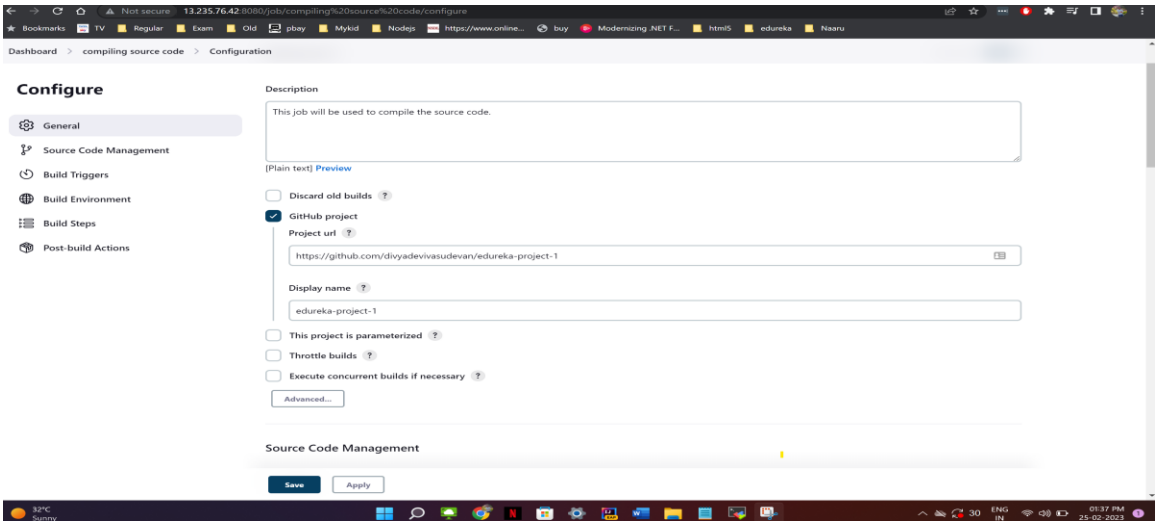
1)compiling source code 2) Test code and 3) Package code

Steps :

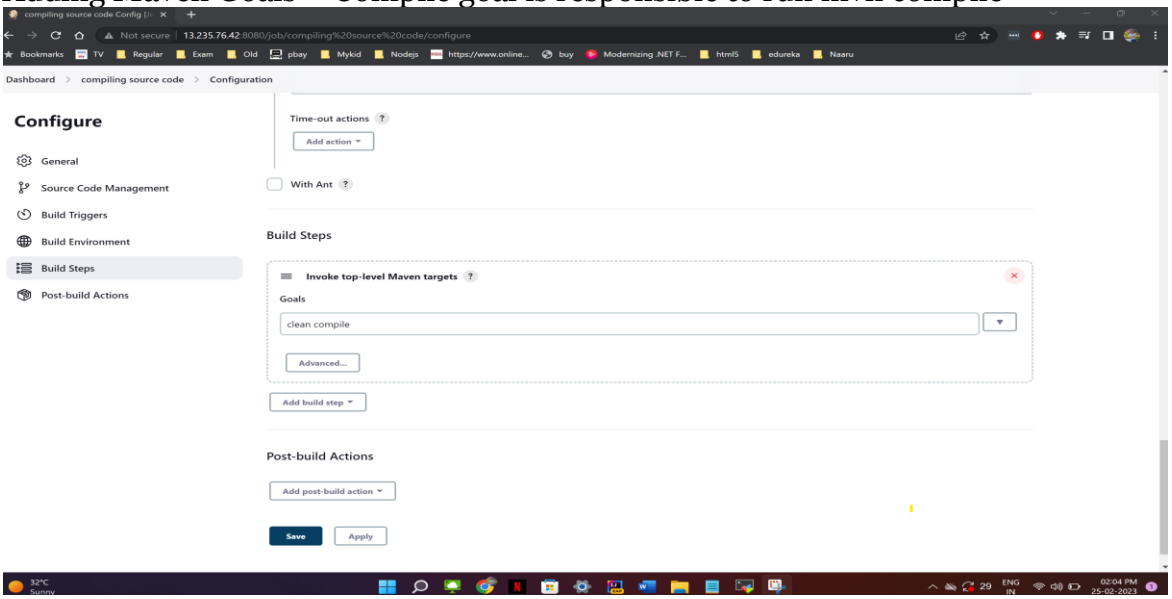
1) Click New Item -> give a name to it Like -> compile Source code
2) Then click Freestyle project and save it.
3) We can then click on configure and add the github url from where the source code has to be pulled and then add the maven goals clean and compile. *I also faced error as my Jenkins was not able to pull the repo, so I went ahead and installed git in the EC2 machine to overcome the connectivity issue. [The details added in the document under GIT heading]*

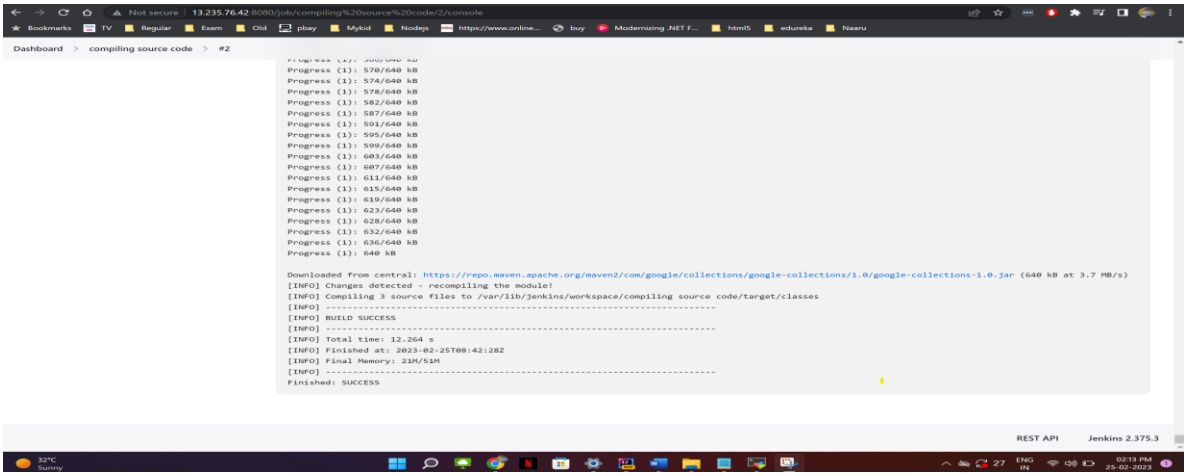


Adding Github details.

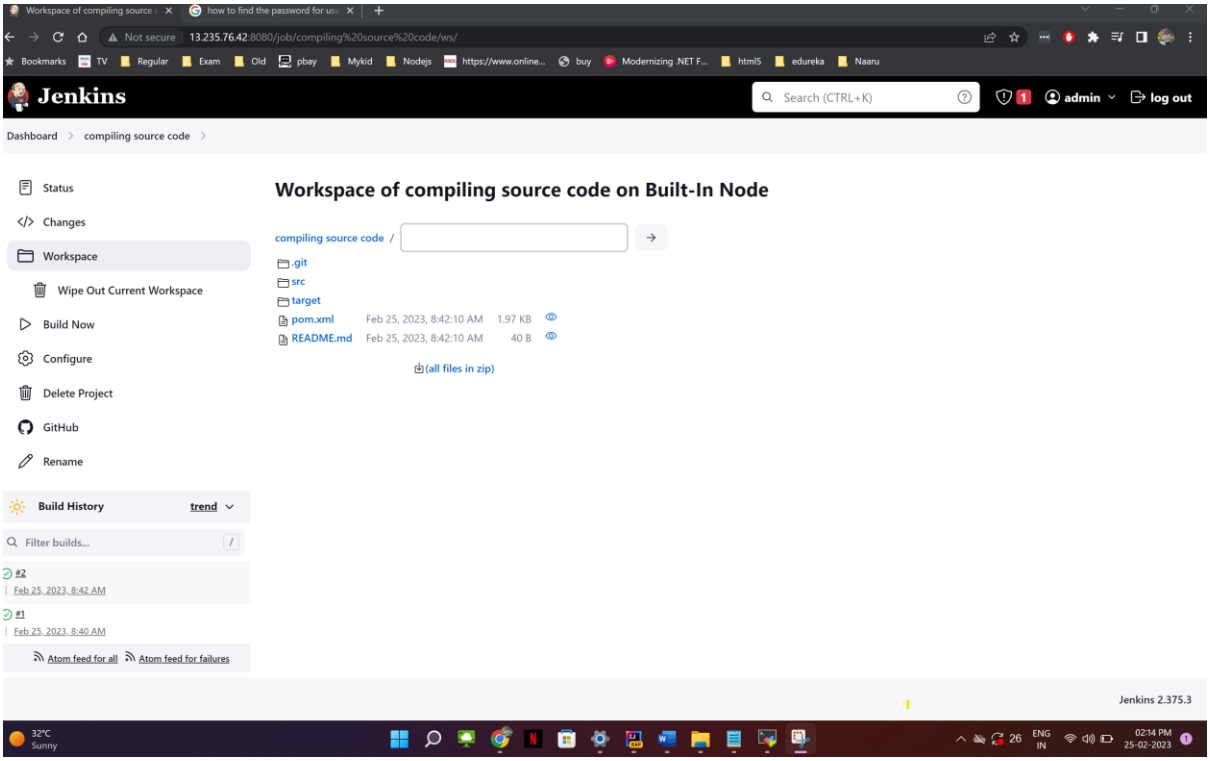


Adding Maven Goals – Compile goal is responsible to run mvn compile





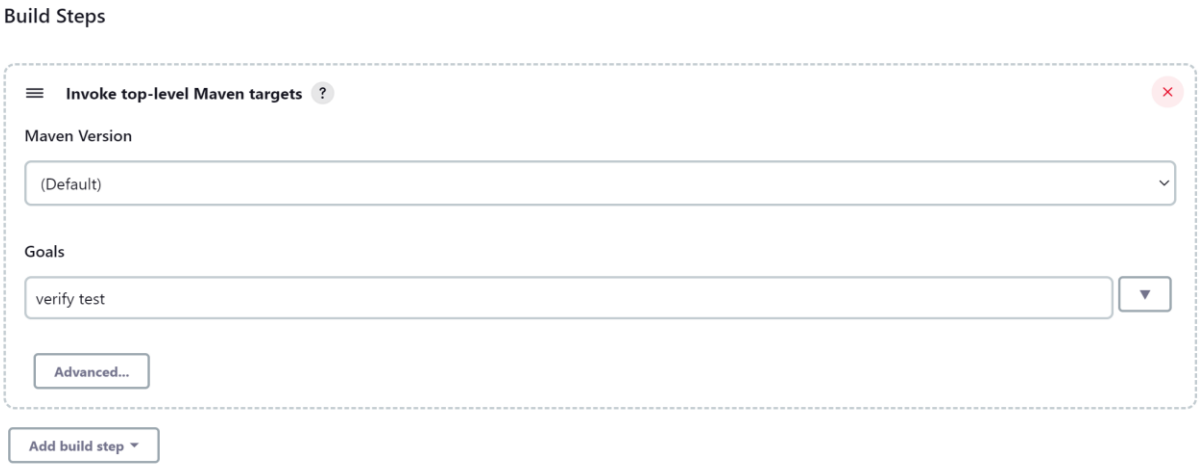
The Compiled source code can be viewed in the target directory of workspace.



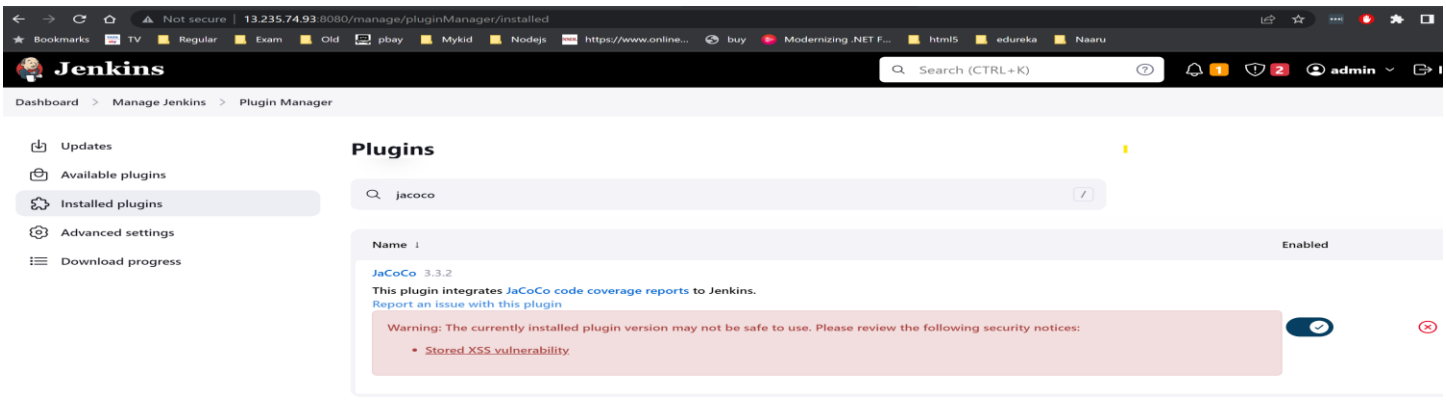
JOB 2 : Review and Test Code

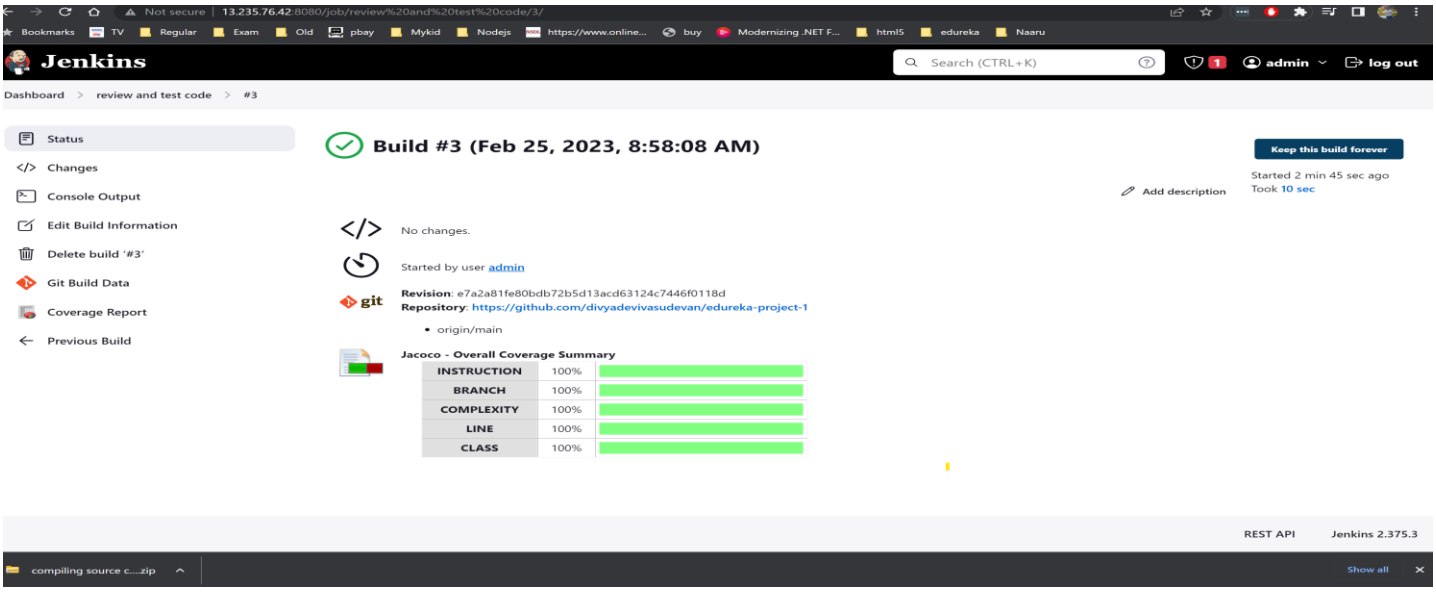
The Github setup is same as the job 1.

The maven goal is verify and test – which is same as our mvn verify and mvn test commands



Also configuring Jacoco reports in the post build step. We need the jacoco plugin



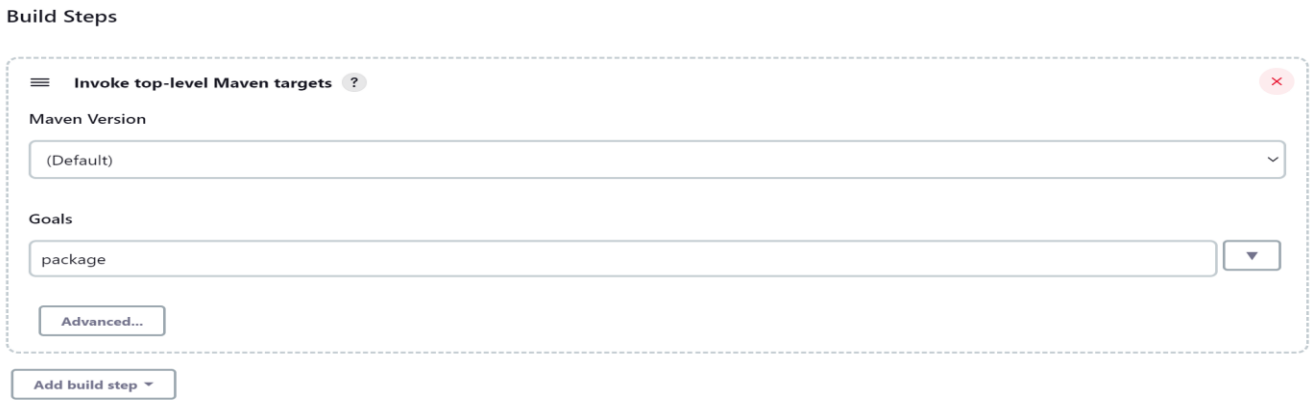


JOB 3 : Package Code

Now package the code into a war file .
The following dependency should be added in the pom.xml if not there already

```
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-war-plugin</artifactId>
   <version>3.3.1</version>
</plugin>
```

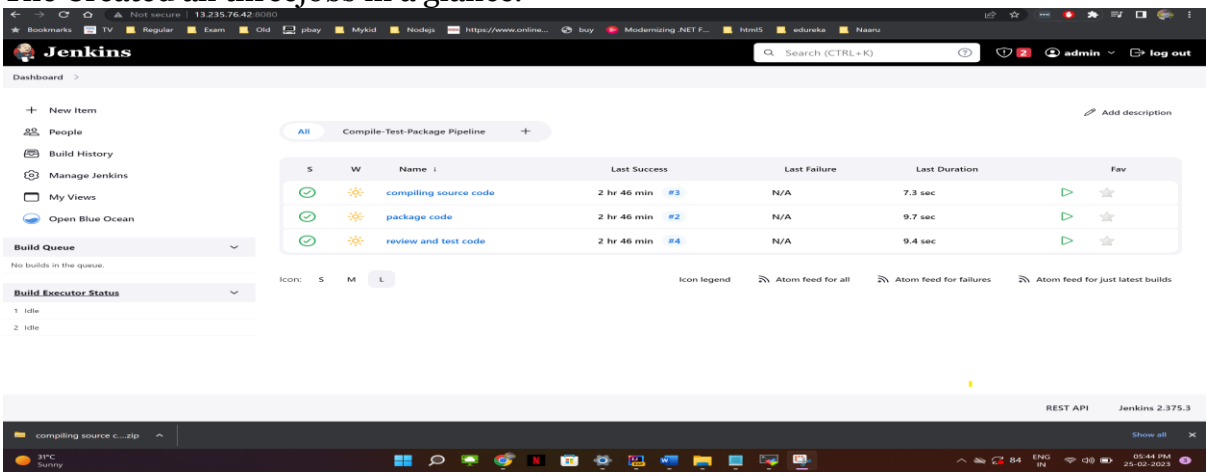
Then repeat the same procedure like above.
Create a new freestyle project and add Github connectivity.
Then in maven goals add package – this is equivalent to command mvn package



Running this will ensure that war is created in the target folder. This can be used to deploy to our tomcat server and launch the application.

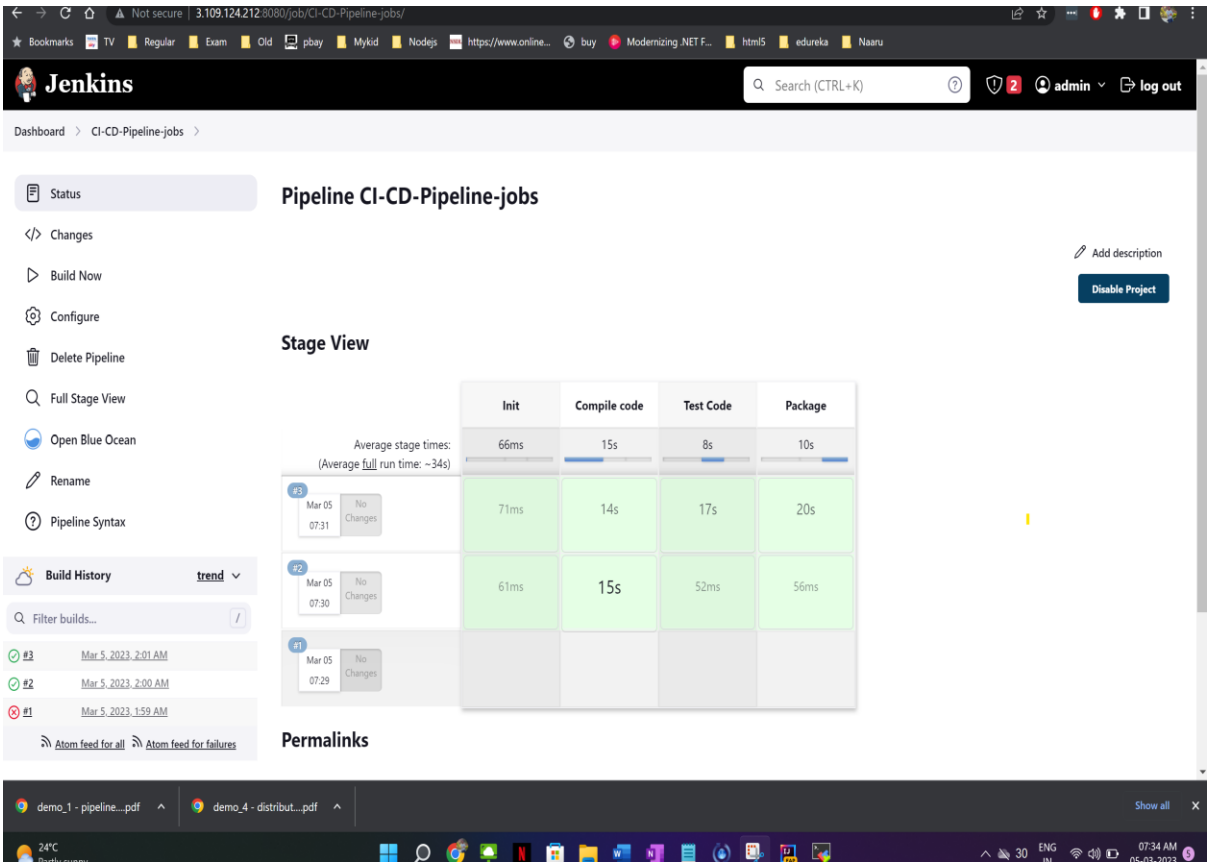
```
[INFO]
[INFO] --- maven-war-plugin:3.2.2:war (default-war) @ ABCtechnologies ---
[INFO] Packaging webapp
[INFO] Assembling webapp [ABCtechnologies] in [/var/lib/jenkins/workspace/package code/target/ABCtechnologies-1.0]
[INFO] Processing war project
[INFO] Copying webapp resources [/var/lib/jenkins/workspace/package code/src/main/webapp]
[INFO] Webapp assembled in [140 msecs]
[INFO] Building war: /var/lib/jenkins/workspace/package code/target/ABCtechnologies-1.0.war
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.6:report (jacoco-site) @ ABCtechnologies ---
[INFO] Loading execution data file /var/lib/jenkins/workspace/package code/target/jacoco.exec
[INFO] Analyzed bundle 'RetailModule' with 2 classes
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ ABCtechnologies ---
[INFO] Installing /var/lib/jenkins/workspace/package code/target/ABCtechnologies-1.0.war to
/var/lib/jenkins/.m2/repository/com/abc/ABCtechnologies/1.0/ABCtechnologies-1.0.war
[INFO] Installing /var/lib/jenkins/workspace/package code/pom.xml to /var/lib/jenkins/.m2/repository/com/abc/ABCtechnologies/1.0/ABCtechnologies-1.0.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 6.673 s
[INFO] Finished at: 2023-03-05T02:02:53Z
[INFO] Final Memory: 17M/84M
[INFO] ------------------------------------------------------------------------
Finished: SUCCESS
```

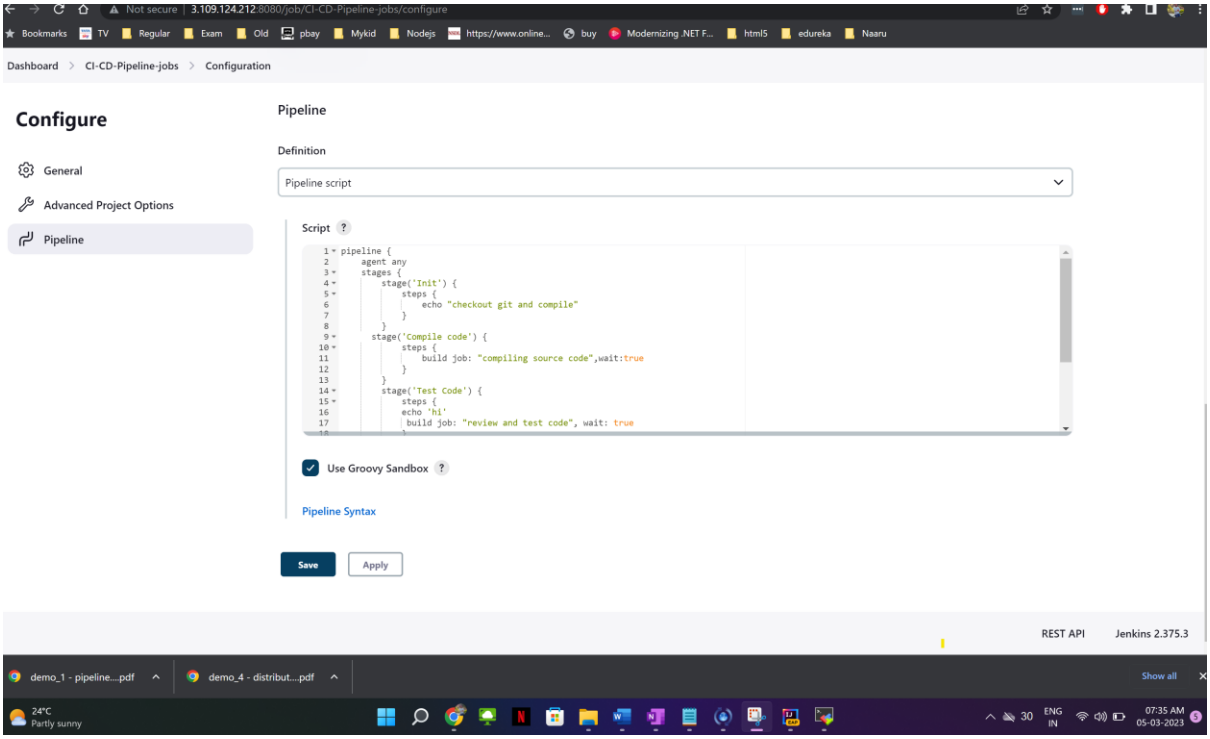
The Created all threejobs in a glance.



The above three jobs can be integrated into a pipeline.

Create new item → Pipeline → provide the suitable name [I have given here CI Cd pipeline]
The the commands to run the three tasks are coming from the pipeline script.





Master-Slave setup for jenkins

Steps -

Installed Java on master node

Installed Jenkins,Java,Git,Maven on master node

Installed java,Git and Maven on slave node

Created a user and ssh keys on slave node

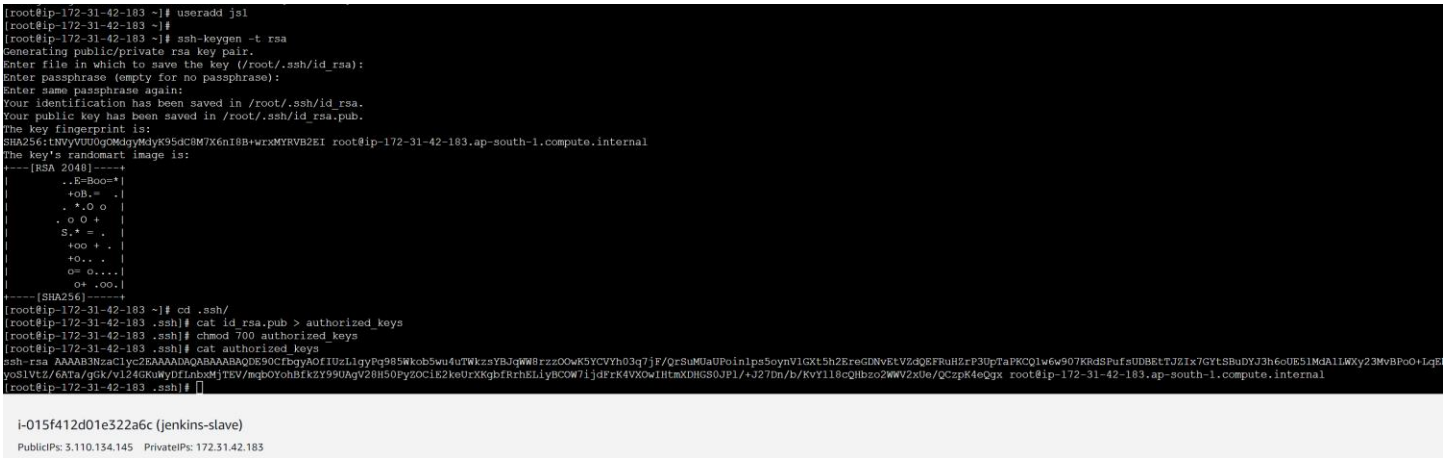
Copy keys on master node and join Slave with MAsster

Create Jenkins File – added slave labels to stages that should be run by slave.

Jenkins Running on master::

```
[root@ip-172-31-46-53 ~]# service jenkins start
Starting jenkins (via systemctl):                          [  OK  ]
[root@ip-172-31-46-53 ~]# systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2023-05-24 15:22:13 UTC; 44s ago
 Main PID: 4788 (java)
    Tasks: 44
   Memory: 410.7M
   CGroup: /system.slice/jenkins.service
           └─4788 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=%C/jenkins/war --httpPort=8080
```

Java installed in slave and created ssh key and user



Copy keys in master

sudo mkdir /var/lib/jenkins/.ssh
sudo -i

chown jenkins:jenkins /var/lib/jenkins/.ssh/
echo "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDE90CfbgyAOfIUzL1gyPq985Wkob5wu4uTWkzsYBJqWW8rzzOOwK5YCVYh03q7jF/QrSu
MUaUPoinlps5oynV1GXt5h2EreGDNvEtVZdQEFRuHZrP3UpTaPKCQ1w6w907KRdSPufsUDBEtTJZIx7GYtSBuDYJ3h6oUE5lMdA1L
WXy23MvBPoO+LqEhnyoSlVtZ/6ATa/gGk/vl24GKuWyDfLnbxMjTEV/mqbOYohBfkZY99UAgV28H50PyZOCiE2keUrXKgbfRrhELiy
BCOW7ijdFrK4VXOwIHtmXDHGS0JPl/+J27Dn/b/KvY1l8cQHbzo2WWV2xUe/QCzpK4eQgx root@ip-172-31-42-183.ap-south-
1.compute.internal" > /var/lib/jenkins/.ssh/known_hosts

cat /var/lib/jenkins/.ssh/known_hosts
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDE90CfbgyAOfIUzL1gyPq985Wkob5wu4uTWkzsYBJqWW8rzzOOwK5YCVYh03q7jF/QrSu
MUaUPoinlps5oynV1GXt5h2EreGDNvEtVZdQEFRuHZrP3UpTaPKCQ1w6w907KRdSPufsUDBEtTJZIx7GYtSBuDYJ3h6oUE5lMdA1L
WXy23MvBPoO+LqEhnyoSlVtZ/6ATa/gGk/vl24GKuWyDfLnbxMjTEV/mqbOYohBfkZY99UAgV28H50PyZOCiE2keUrXKgbfRrhELiy
BCOW7ijdFrK4VXOwIHtmXDHGS0JPl/+J27Dn/b/KvY1l8cQHbzo2WWV2xUe/QCzpK4eQgx root@ip-172-31-42-183.ap-south-
1.compute.internal

6. Join slave node to master

To join the Jenkins slave node to Jenkins Master, performed below steps -

Select Build Executor Status > New Node > Type - Permanent

Name - jenkins-slave1

Description - jenkins-slave1

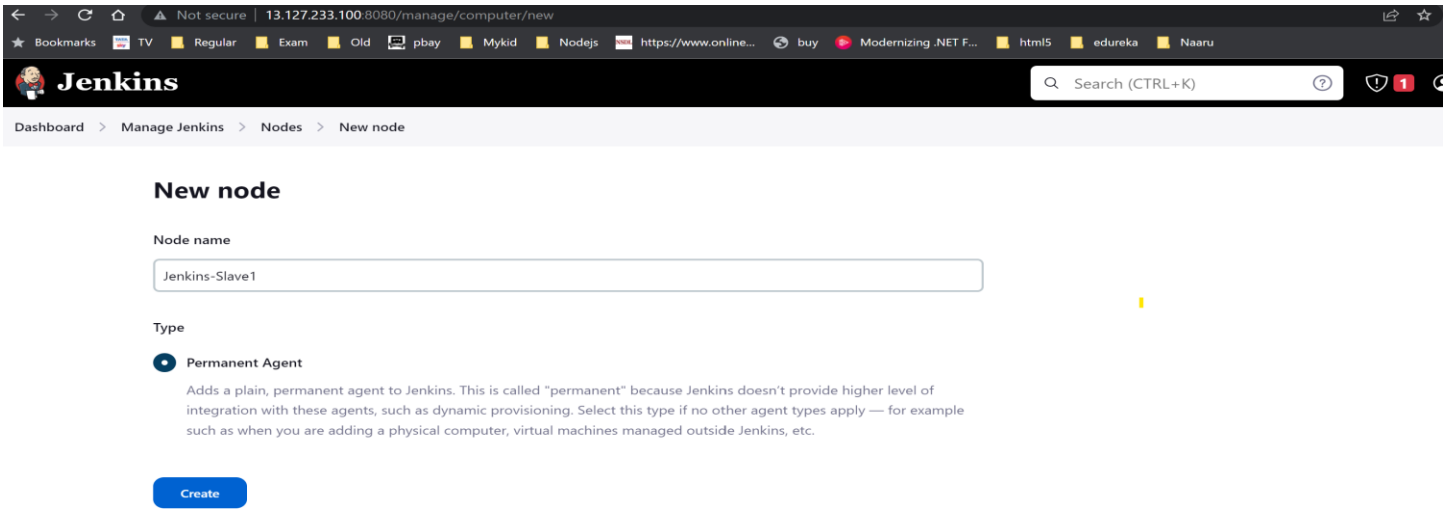
Number of executors - 1

Remote root directory - /home/ec2-user

Labels - jenkins-slave1

Usage - Use this mode as much as possible

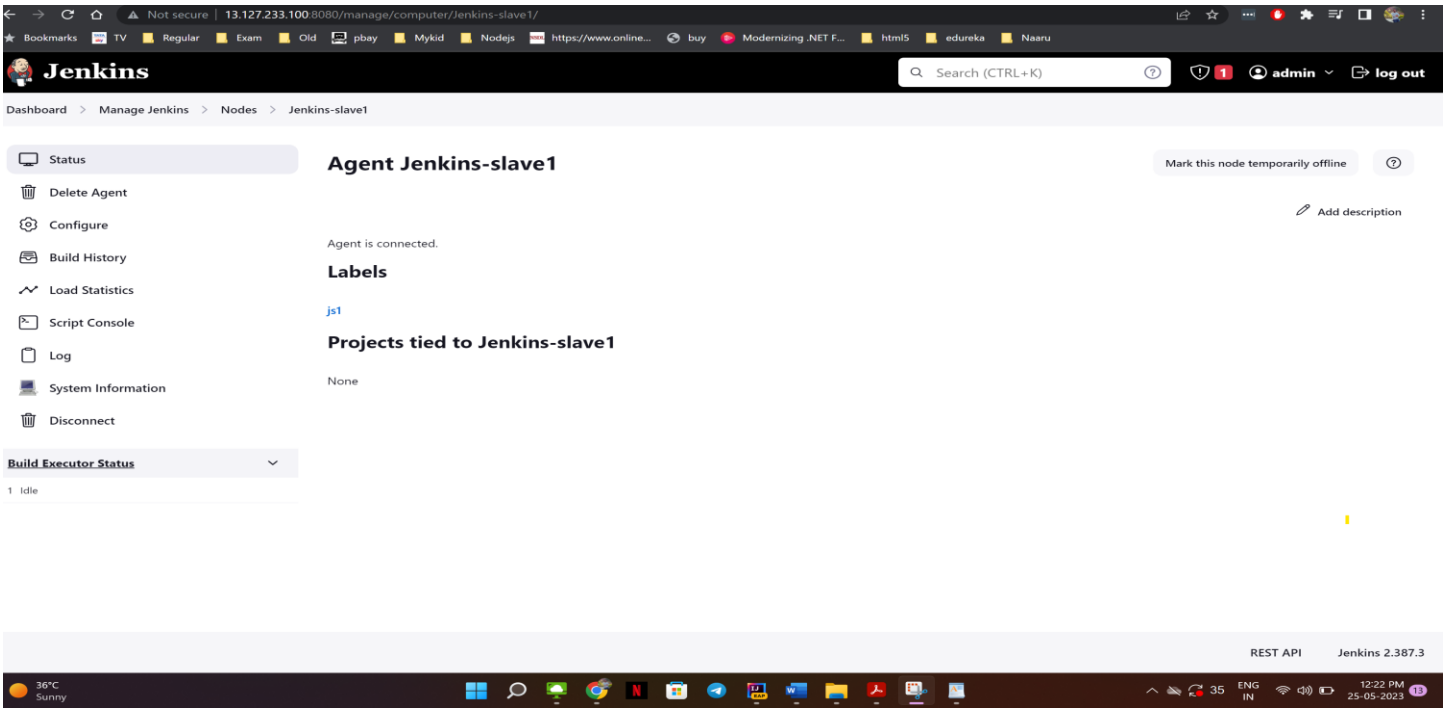
Launch method - Launch agents via SSH



Gave the Host – IP and Credentials - use ssh username / private key options

Host Key Verification Strategy - Know hosts key strategy

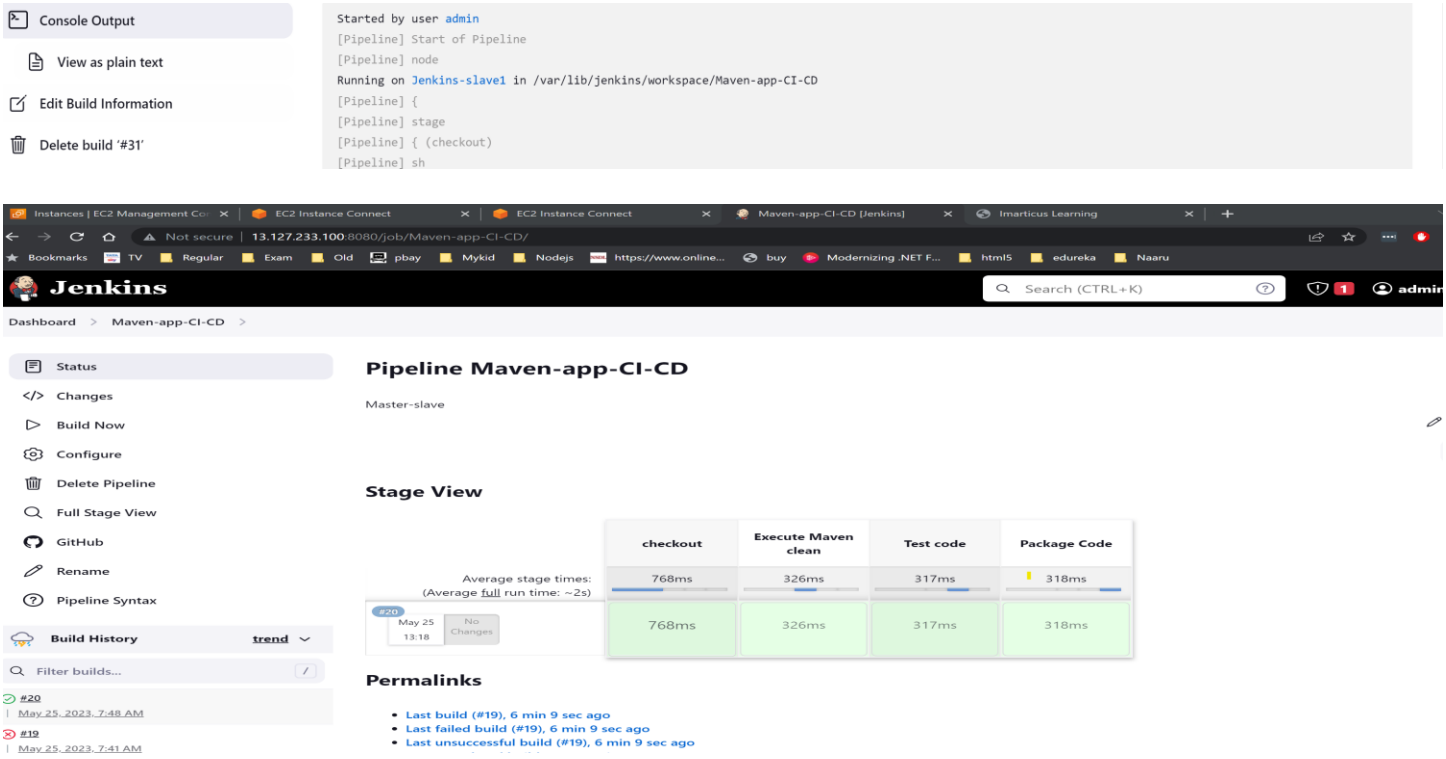
Save and check that new slave node is getting connected. Once connected wit will display like below.



Executed the below pipeline script.

```
pipeline {
   agent {label 'js1'}

 stages {
    stage('checkout') {
        steps {
            git branch: 'main', url: 'https://github.com/divyadevivasudevan/edureka-project-1.git'

        }
      }
  stage('Execute Maven clean') {
        steps {
            sh 'mvn  clean'
```

```
        }
      }

stage('Test code') {
        steps {

            sh 'mvn test verify '
        }
      }
 stage('Package Code') {

        steps {
        agent {label 'js1'}
          sh 'mvn package '

        }
      }
  }
}
```



Commited the pipeline script to the repo. File name : JenkinsFile-masterslave.

---------- END of TASK 2----------

Task 3: Write a Docker file. Create an Image and container on the Docker host. Integrate docker host with Jenkins. Create CI/CD job on Jenkins to build and deploy on a container.

1. Enhance the package job created in step 1 of task 2 to create a docker image.
2. In the Docker image, add code to move the war file to the Tomcat server and build the image.

What is docker?

Docker is a set of platforms as a service product that use OS-level virtualization to deliver software in packages called containers. The service has both free and premium tiers. The software that hosts the containers is called Docker Engine.

As pre-requisite :

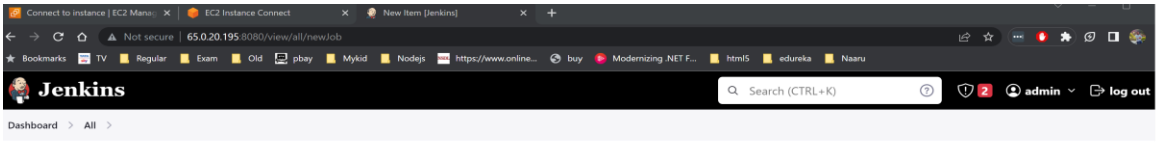
Created a docker-hub account.
Installed docker in my VM.

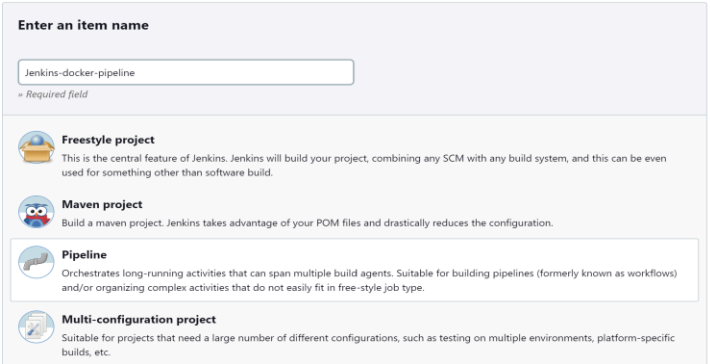
The script for the same is commited with the file name :: installdocker.sh

```
sudo -i
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo
https://download.docker.com/linux/rhel/9/x86_64/stable/repodata/repomd.xml
sudo yum install docker
sudo systemctl start docker
sudo systemctl enable docker
```

CI-CD pipeline creation.

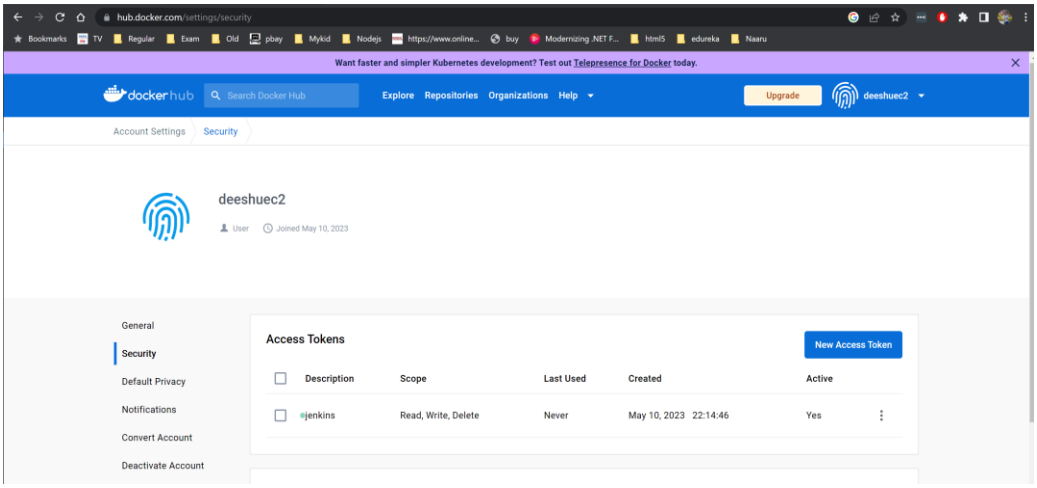
1. Create new item → Pipeline → provide a name .



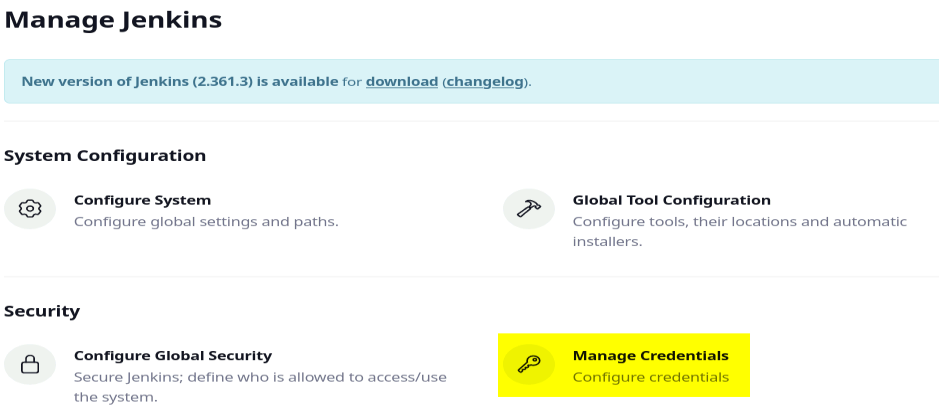


2. Add Docker Hub access token into Jenkins

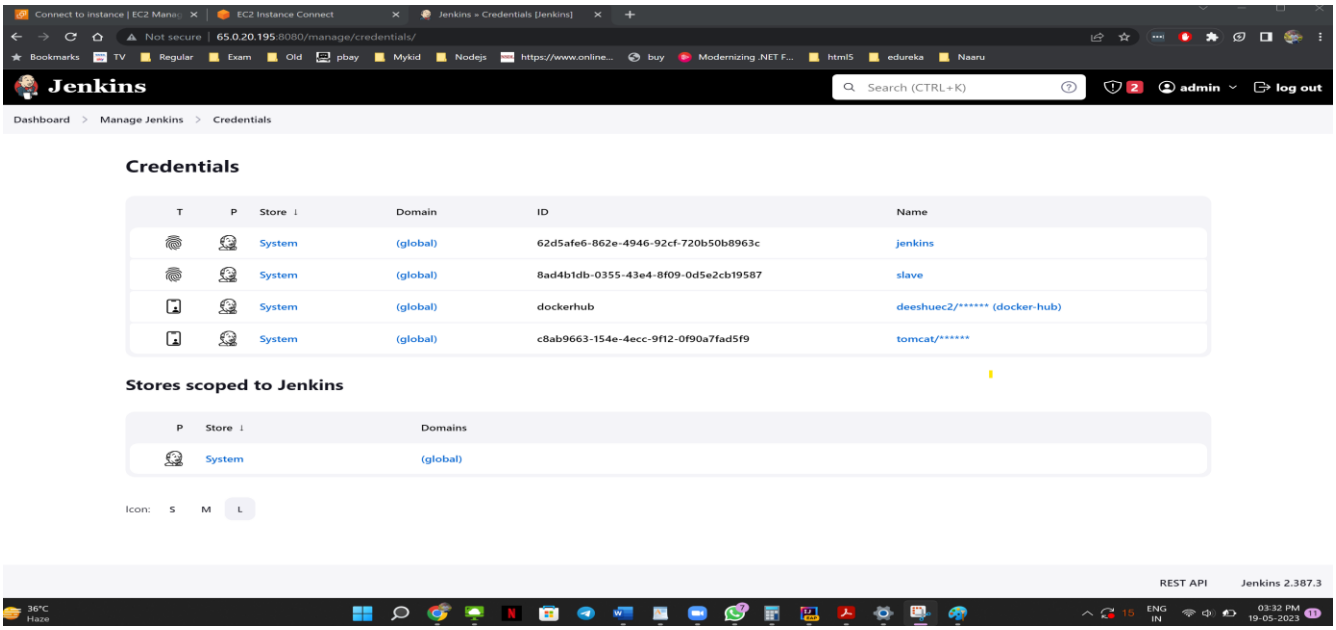
a. Create new Access token on your DockerHub account under Security section.



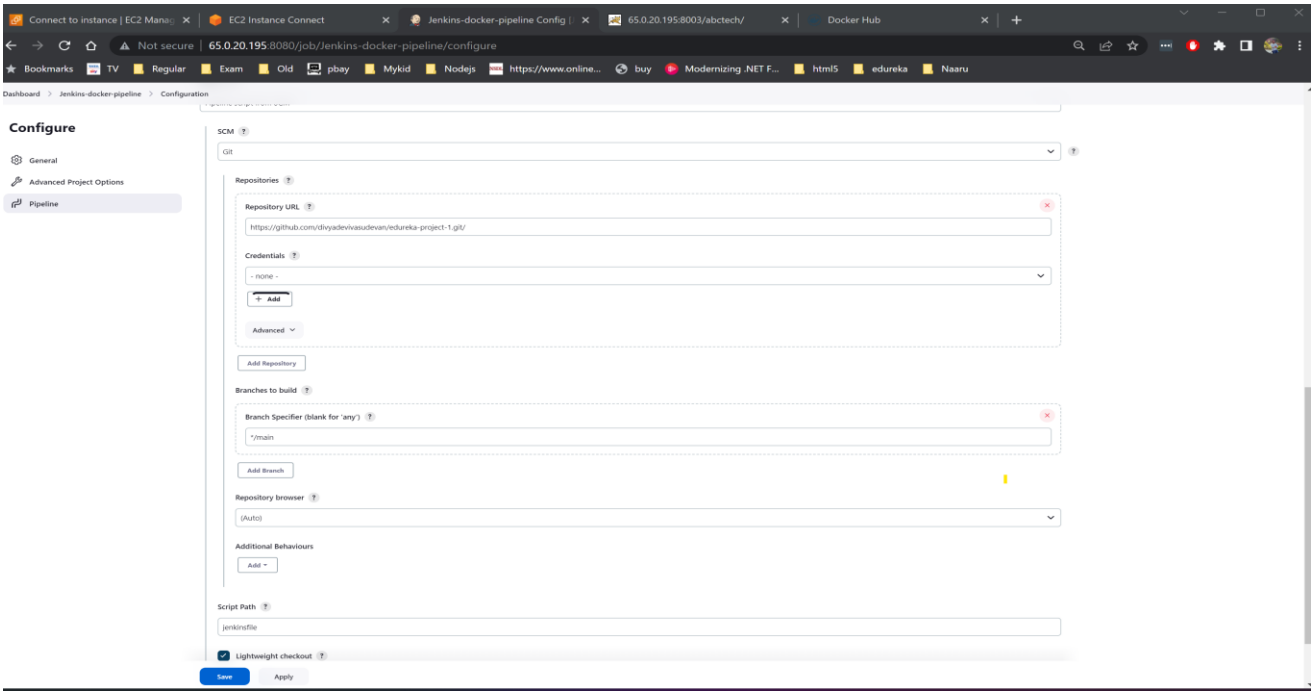
b. Open Manage Jenkins –> Mange Credentials and Add the System Credentials into Jenkins



.



2. Go to Pipeline section add definition pipeline script from SCM and setup the other required config like below. Mainly the Jenkins filename / or path
   Branch to take from.



   What is the Jenkins pipeline stages that we have written?
   a. We will checkout our git repo.
   b. Run maven clean install and package it to war.
   c. Run docker build to create a image and tag.
   d. Login to Docker hub and push the image.
   e. Start the container in 8003 port number.

While doing the above I faced errors:

1) Was my jenkinsfilename I gave it wrong -> added a typo
2) Login was not successful as I tried with DockerHub as the credential id, however I had it as dockerhub in the Jenkins credential id

These silly typos, made me to run the pipeline multiple times.

We Will also need a dockerfile that will do the following :

1) Install tomcat
2) Install java
3) Copy our war file to the webapps
4) And run the web server for us in the container

So we have now successfully used Jenkins and docker -> compiled , packaged the code and copied the war to tomcat and run the container and were able to access the web application running as a container using http://EC2-server-IP:8003/abctech/ - Code snippets and screenshot for these are in the below pages.

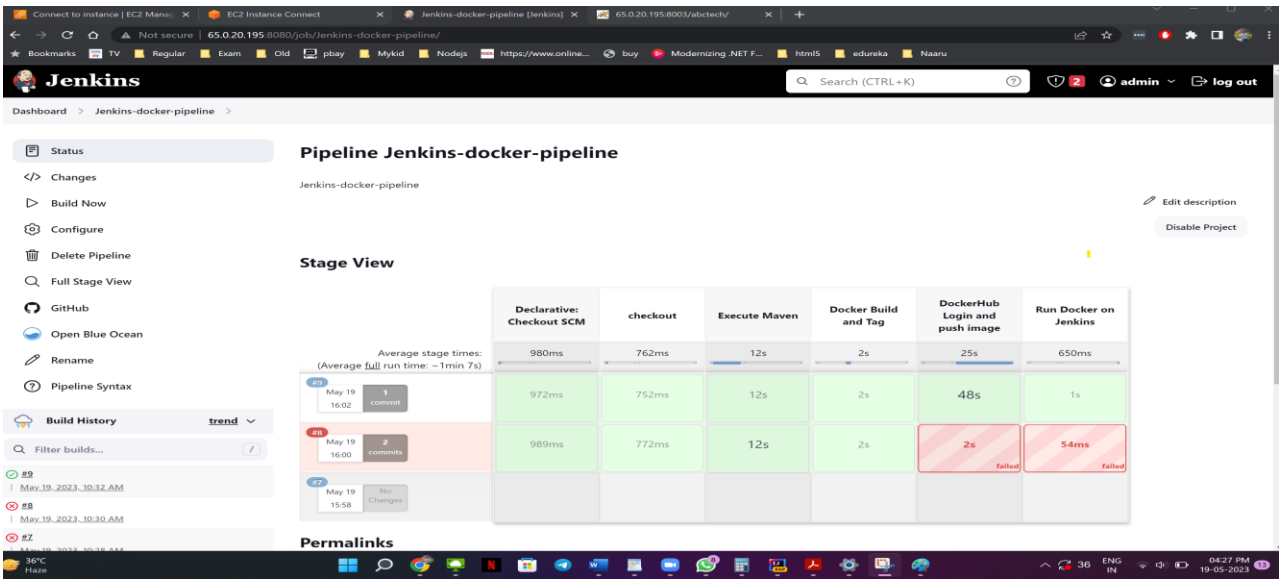
Jenkinsfile

```
pipeline {
    agent any

 stages {
      stage('checkout') {
           steps {

                git branch: 'main', url: 'https://github.com/divyadevivasudevan/edureka-
project-1.git'

           }
         }
  stage('Execute Maven') {
           steps {

                sh 'mvn  clean package'
                sh 'echo package done'
                sh  'mv target/*.war target/abctech.war'
           }
         }

stage('Docker Build and Tag') {
           steps {

                sh 'docker build -t abctechapp:latest .'
                sh 'docker tag abctechapp deeshuec2/abctechapp:latest'


           }
         }

     stage('DockerHub Login and push image') {
          steps {
            //sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u
$DOCKERHUB_CREDENTIALS_USR --password-stdin'

            withCredentials([usernamePassword(credentialsId: 'dockerhub',
passwordVariable: 'dockerhubPassword', usernameVariable: 'dockerhubUser')]) {
                       sh "docker login -u ${env.dockerHubUser} -p
${env.dockerHubPassword}"
                      sh "docker push deeshuec2/abctechapp:latest"
           }
           }
         }


 stage('Run Docker on Jenkins') {

            steps {
               // sh "docker -H ssh://jenkins@172.31.28.25 run -d -p 8003:8080
deeshuec2/samplewebapp"
               sh "docker run -d -p 8003:8080 deeshuec2/abctechapp"

            }
         }
    }
 }
```

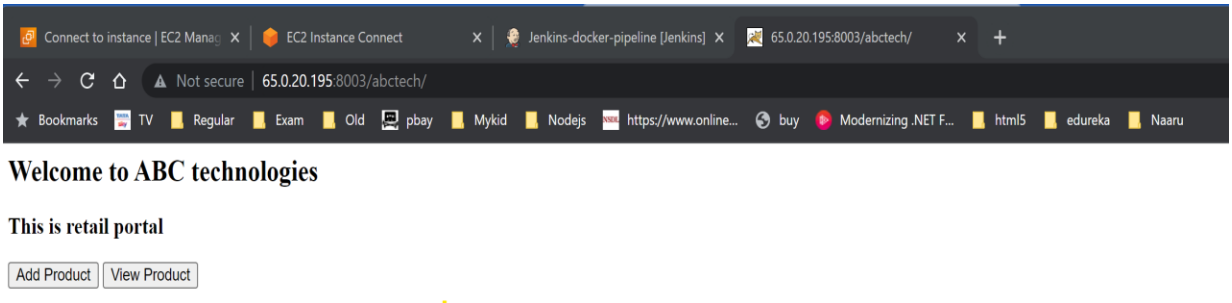
3. Add DockerFile with following commands.

```
FROM docker.io/library/ubuntu:18.04
RUN apt-get -y update && apt-get -y upgrade
RUN apt-get -y install openjdk-8-jdk wget
RUN mkdir /usr/local/tomcat
ADD https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.75/bin/apache-tomcat-9.0.75.tar.gz
/tmp/apache-tomcat-9.0.75.tar.gz
RUN cd /tmp &&  tar xvfz apache-tomcat-9.0.75.tar.gz
RUN cp -Rv /tmp/apache-tomcat-9.0.75/* /usr/local/tomcat/
ADD **/*.war /usr/local/tomcat/webapps
EXPOSE 8089
CMD /usr/local/tomcat/bin/catalina.sh run
```

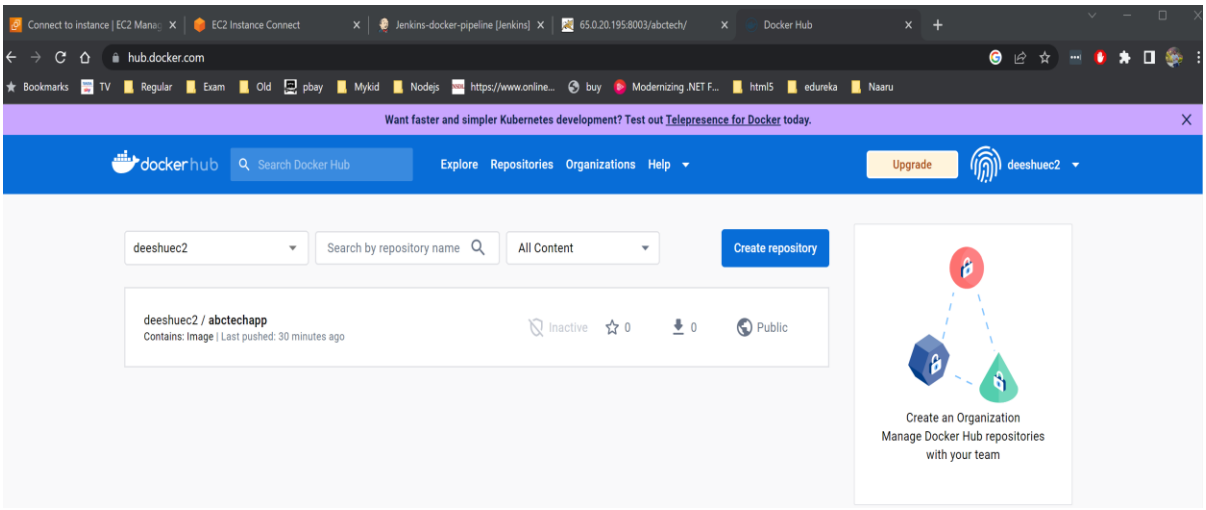
4. Screenshot of the various stages in build.



Access the web application running as a container using http://EC2-server-IP:8003/abctech/ you should see the following web application.



We can also verify that our docker image was pushed to the docker hub.



----------END of TASK 3----------

Task 4: Integrate the Docker host with Ansible. Write an Ansible playbook to create an image and create a continuer. Integrate Ansible with Jenkins. Deploy Ansible-playbook. CI/CD job to build code on ansible and deploy it on docker container

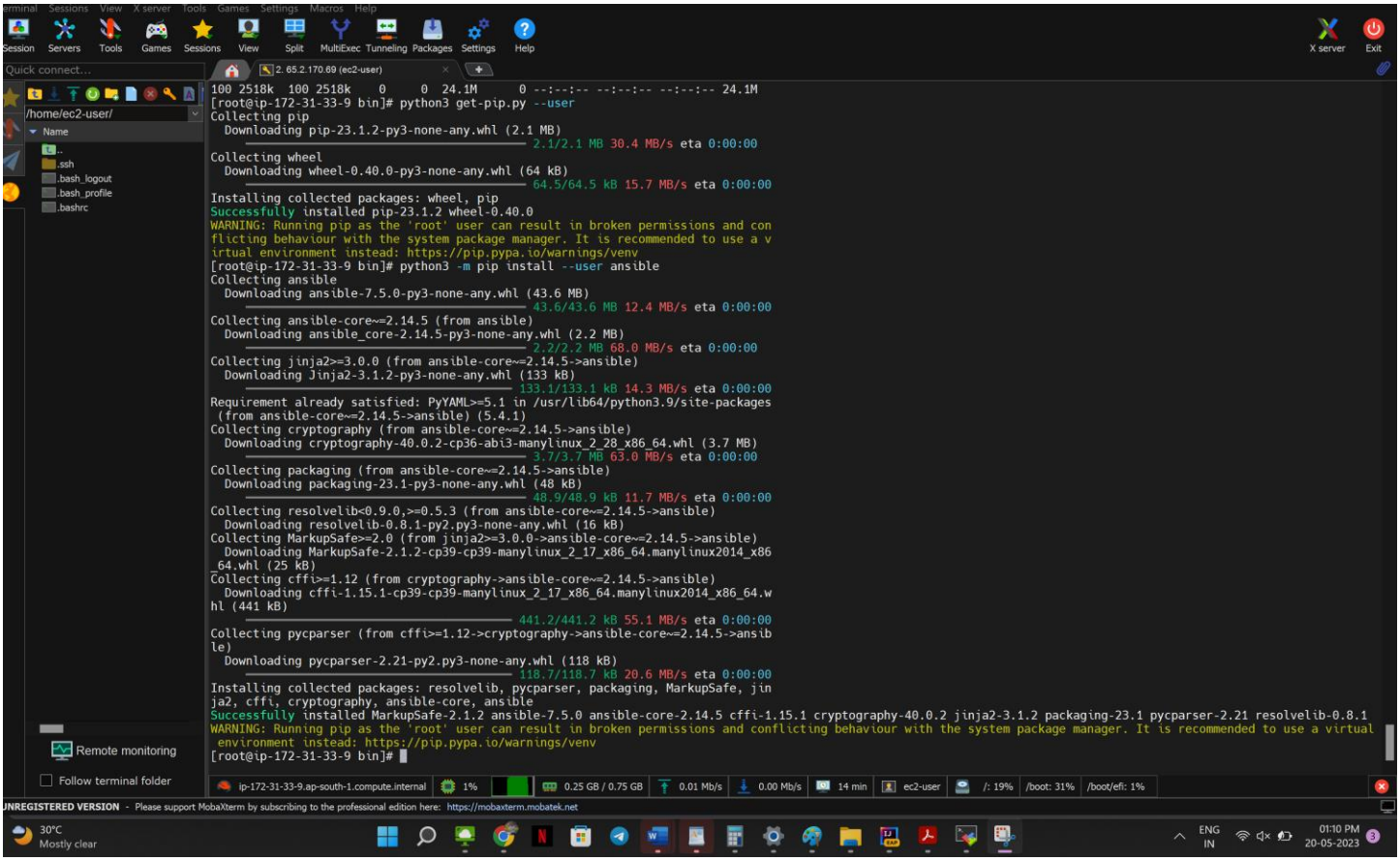
1. Deploy Artifacts on Kubernetes
2. Write pod, service, and deployment manifest file
3. Integrate Kubernetes with Ansible
4. Ansible playbook to create deployment and service

What is ansible?

> Ansible is a radically simple IT automation system. It handles configuration-management, application deployment, cloud provisioning, ad-hoc task-execution, and multinode orchestration - including trivializing things like zero-downtime rolling updates with load balancers.

Installing Ansible.

sudo yum install python
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python3 get-pip.py –user
python3 -m pip install --user ansible
ansible --version



Verifying the ansible version:

```
[root@ip-172-31-33-9 bin]# ansible --version
ansible [core 2.14.5]
  config file = None
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /root/.local/lib/python3.9/site-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /root/.local/bin/ansible
  python version = 3.9.16 (main, Dec  8 2022, 00:00:00) [GCC 11.3.1 20221121 (Red Hat 11.3.1-4)] (/bin/python3)
  jinja version = 3.1.2
  libyaml = True
[root@ip-172-31-33-9 bin]#
```

Integrating Docker host with Ansible

Add Ansible Server user

$ sudo su -
$ useradd ansadmin
$ passwd ansadmin
Changing password for user ansadmin
New password:
Retype new password:
passwd: all authentication tokens updated successfully.

Grant sudo Access to ansible user

$ visudo

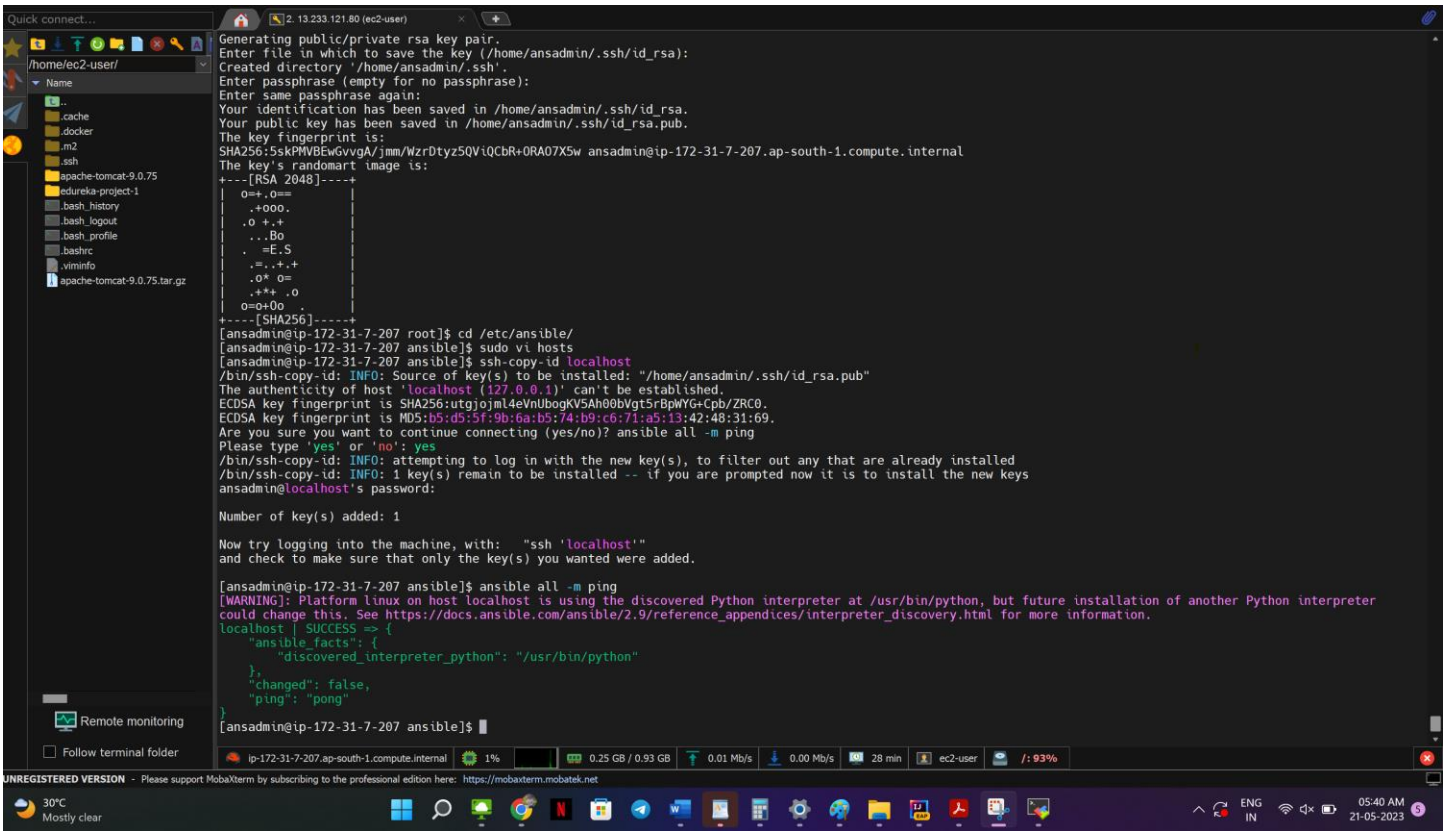
ansadmin ALL=(ALL) NOPASSWD: ALL

Setting up Password Authentication

```
$ vi /etc/ssh/sshd_config # To disable tunneled clear text passwords, change to no here! PasswordAuthentication yes #PermitEmptyPasswords no #PasswordAuthentication no
```

Creating SSHKEY

Ssh-keygen



In Ansible server :: Create hosts file in /etc/ansible

$ sudo vi hosts
target server private-ip
localhost
Before the test localhost Keygen id copy

$ ssh-copy-id localhost
PING TEST

$ ansible all -m ping

```
localhost | SUCCESS => {
   "ansible_facts": {
      "discovered_interpreter_python": "/usr/bin/python"
   },
   "changed": false,
   "ping": "pong"
}
```

Creating Ansible Playbook

$vi ansible-docker.yml

```
---
- hosts: localhost
  become: true
  tasks:
    - name: stop if we have old docker container
      command: docker stop devops-container
      ignore_errors: yes

    - name: remove stopped docker container
      command: docker rm devops-container
      ignore_errors: yes

    - name: remove current docker image
      command: docker rmi jayjodev/devops-image
      ignore_errors: yes

    - name: pull docker image from dockerhub
      command: docker pull deeshuec2/abctechapp:latest

    - name: creating docker image
      command: docker run -d --name devops-container -p 8010:8080 deeshuec2/abctechapp
```

```
[ansadmin@ip-172-31-7-207 docker]$ sudo vi ansible-docker-hub-pull-deploy.yml
[ansadmin@ip-172-31-7-207 docker]$ ansible-playbook -i hosts  ansible-docker-hub-pull-deploy.yml
[WARNING]: Unable to parse /opt/docker/hosts as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [localhost] ***************************************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************************
ok: [localhost]

TASK [stop if we have old docker container] *************************************************************************************************************************************
fatal: [localhost]: FAILED! => {"changed": true, "cmd": ["docker", "stop", "devops-container"], "delta": "0:00:00.090012", "end": "2023-05-21 08:15:27.656446", "msg":
 "non-zero return code", "rc": 1, "start": "2023-05-21 08:15:27.566434", "stderr": "Error response from daemon: No such container: devops-container", "stderr_lines":
["Error response from daemon: No such container: devops-container"], "stdout": "", "stdout_lines": []}
...ignoring

TASK [remove stopped docker container] ******************************************************************************************************************************************
fatal: [localhost]: FAILED! => {"changed": true, "cmd": ["docker", "rm", "devops-container"], "delta": "0:00:00.065394", "end": "2023-05-21 08:15:27.987619", "msg": "
non-zero return code", "rc": 1, "start": "2023-05-21 08:15:27.922225", "stderr": "Error: No such container: devops-container", "stderr_lines": ["Error: No such contai
ner: devops-container"], "stdout": "", "stdout_lines": []}
...ignoring

TASK [remove current docker image] **********************************************************************************************************************************************
fatal: [localhost]: FAILED! => {"changed": true, "cmd": ["docker", "rmi", "jayjodev/devops-image"], "delta": "0:00:00.067644", "end": "2023-05-21 08:15:28.317993", "m
sg": "non-zero return code", "rc": 1, "start": "2023-05-21 08:15:28.250349", "stderr": "Error: No such image: jayjodev/devops-image", "stderr_lines": ["Error: No such
 image: jayjodev/devops-image"], "stdout": "", "stdout_lines": []}
...ignoring

TASK [pull docker image from dockerhub] *****************************************************************************************************************************************
changed: [localhost]

TASK [creating docker image] ****************************************************************************************************************************************************
changed: [localhost]

PLAY RECAP **********************************************************************************************************************************************************************
localhost                  : ok=6    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=3

[ansadmin@ip-172-31-7-207 docker]$ docker images
REPOSITORY              TAG         IMAGE ID       CREATED         SIZE
abctech                 latest      5399986d5f6f   2 hours ago     560MB
deeshuec2/abctechapp    latest      c97961eef911   42 hours ago    560MB
abctechapp              latest      c97961eef911   42 hours ago    560MB
deeshuec2/abctechapp    <none>      e330a037c3b5   46 hours ago    560MB
<none>                  <none>      e4028a00d387   46 hours ago    560MB
<none>                  <none>      22620133d4d2   46 hours ago    560MB
```

ip-172-31-7-207.ap-south-1.compute.internal | 1% | 0.66 GB / 0.93 GB | 0.01 Mb/s | 0.00 Mb/s | 8 hours | ec2-user | /: 94%

baXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

ENG IN 01:47 PM 21-05-2023

Verify Docker container

```
[ansadmin@ip-172-31-7-207 docker]$ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS          PORTS                                                NAMES
a7f86823502e   deeshuec2/abctechapp   "/bin/sh -c '/usr/lo…"   55 seconds ago   Up 54 seconds   8089/tcp, 0.0.0.0:8010->8080/tcp, :::8010->8080/tcp  devops-container
062ad6914109   abctechapp             "/bin/sh -c '/usr/lo…"   9 minutes ago    Up 9 minutes    8089/tcp, 0.0.0.0:8003->8080/tcp, :::8003->8080/tcp  abc-retail-cart
[ansadmin@ip-172-31-7-207 docker]$
```

ip-172-31-7-207.ap-south-1.compute.internal | 1% | 0.66 GB / 0.93 GB | 0.01 Mb/s | 0.00 Mb/s | 8 hours | ec2-user | /: 94%

obaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

ENG IN 01:48 PM 21-05-2023

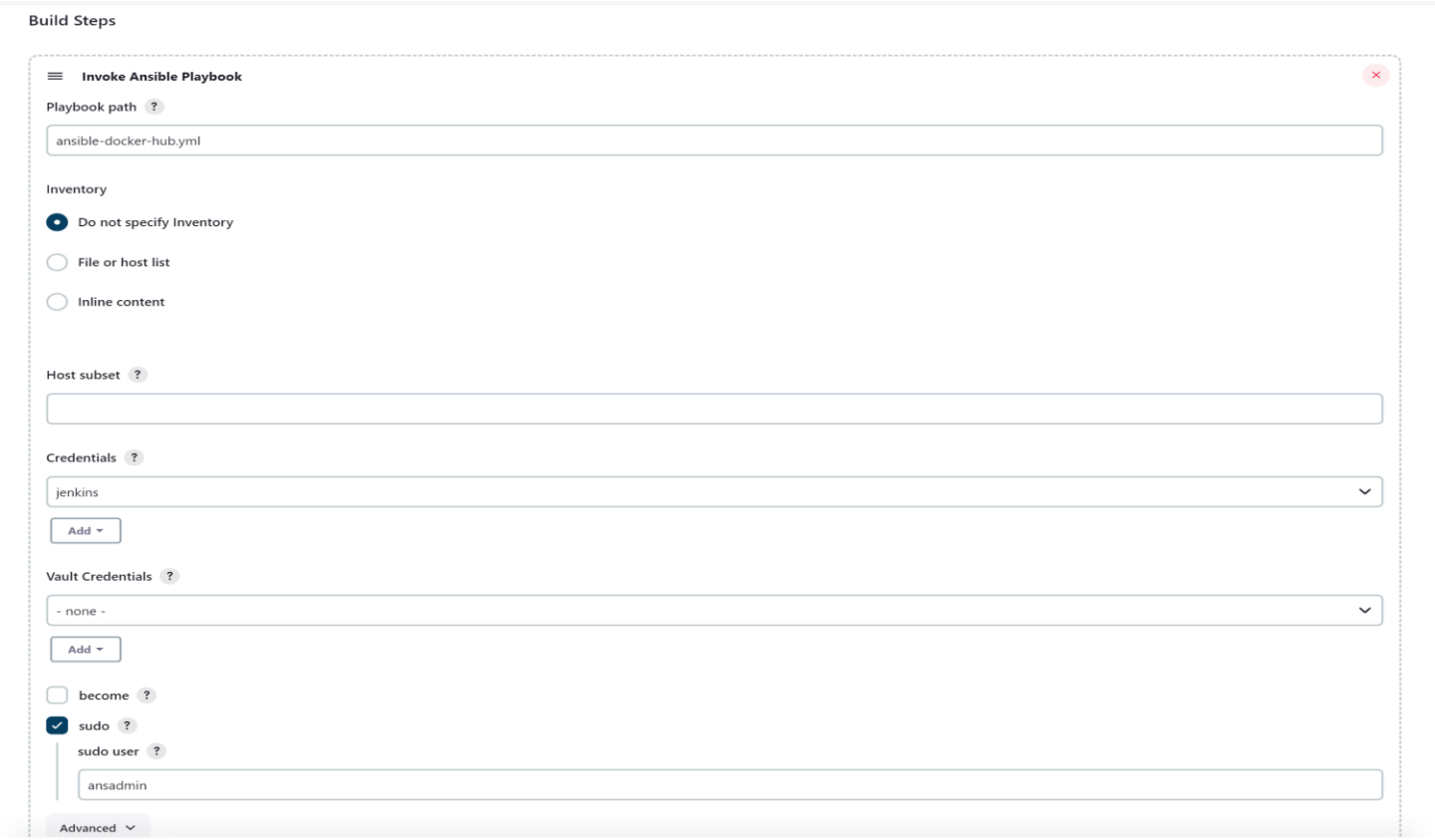
Ansible and Jenkins Integration

Did this with the help of Ansible Plugin.

We can see the Invoke Ansible Playbook option in the Build Environment section but we need to configure Ansible path for Jenkins.

Go to Manage Jenkins > Global Tool Configuration > It will display Ansible on the list

Then created a Freestyle project.
Configured Ansible Plugin.



What is Kubernetes?

Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management. Google originally designed Kubernetes, but the Cloud Native Computing Foundation now maintains the project.

Deploy Artifacts on Kubernetes.

Applications can be installed in Kubernetes using Helm charts. Helm charts are packages that contain all the information that Kubernetes needs to know for managing a specific application within the cluster.

There are two different interfaces from which you can manage the resources on your cluster:

Kubernetes command line interface: kubectl
Kubernetes web-based user interface: Dashboard

Installing Kubernetes

---

1. Download the latest release with the command:

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl
```

   To download a specific version, replace the `$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)` portion of the command with the specific version.

   For example, to download version v1.7.0 on Linux, type:

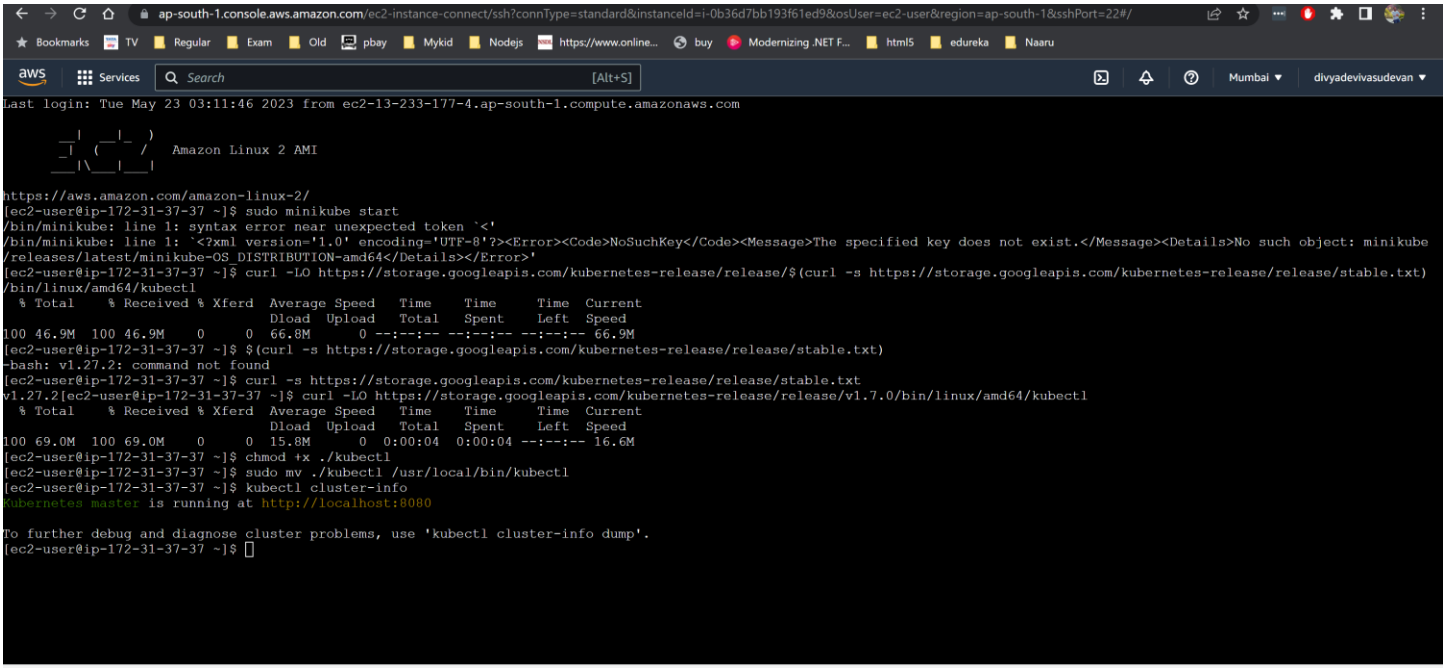
```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.7.0/bin/linux/amd64/kubectl
```

3. Make the kubectl binary executable.

```
chmod +x ./kubectl
```

5. Move the binary in to your PATH.

```
sudo mv ./kubectl /usr/local/bin/kubectl
```



Setting up Kubernetes Cluster

---

cat <<EOF > /etc/yum.repos.d/Kubernetes.repo
[Kubernetes]
name=Kubernetes
baseurl= https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=0
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF

cat <<EOF > /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables=1
net.bridge.bridge-nf-call-iptables=1
EOF

sysctl --system

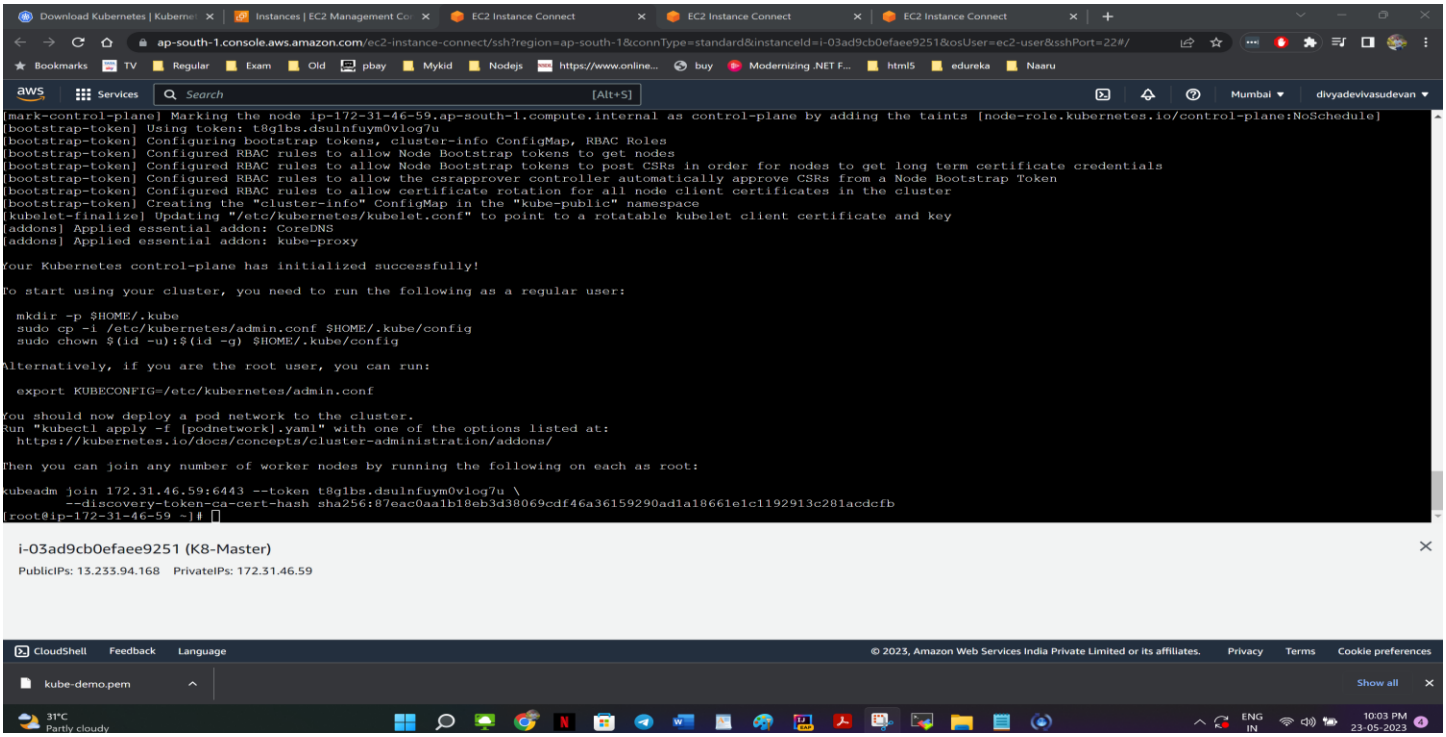
yum install -y kubeadm-1.21.3 kubelet-1.21.3 kubectl-1.21.3 --disableexcludes=kubernetes
yum install -y kubeadm-1.25 kubelet-1.25 kubectl-1.25 --disableexcludes=kubernetes

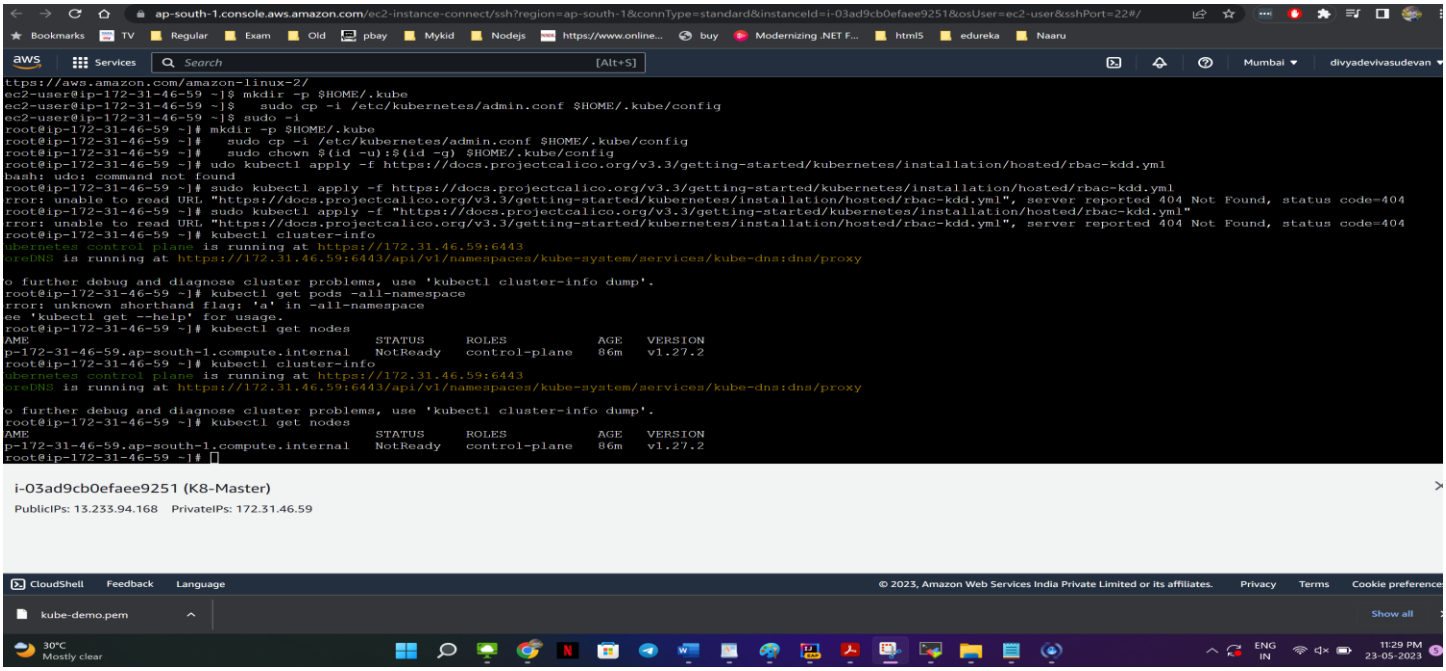
systemctl enable kubelet

systemctl start kubelet

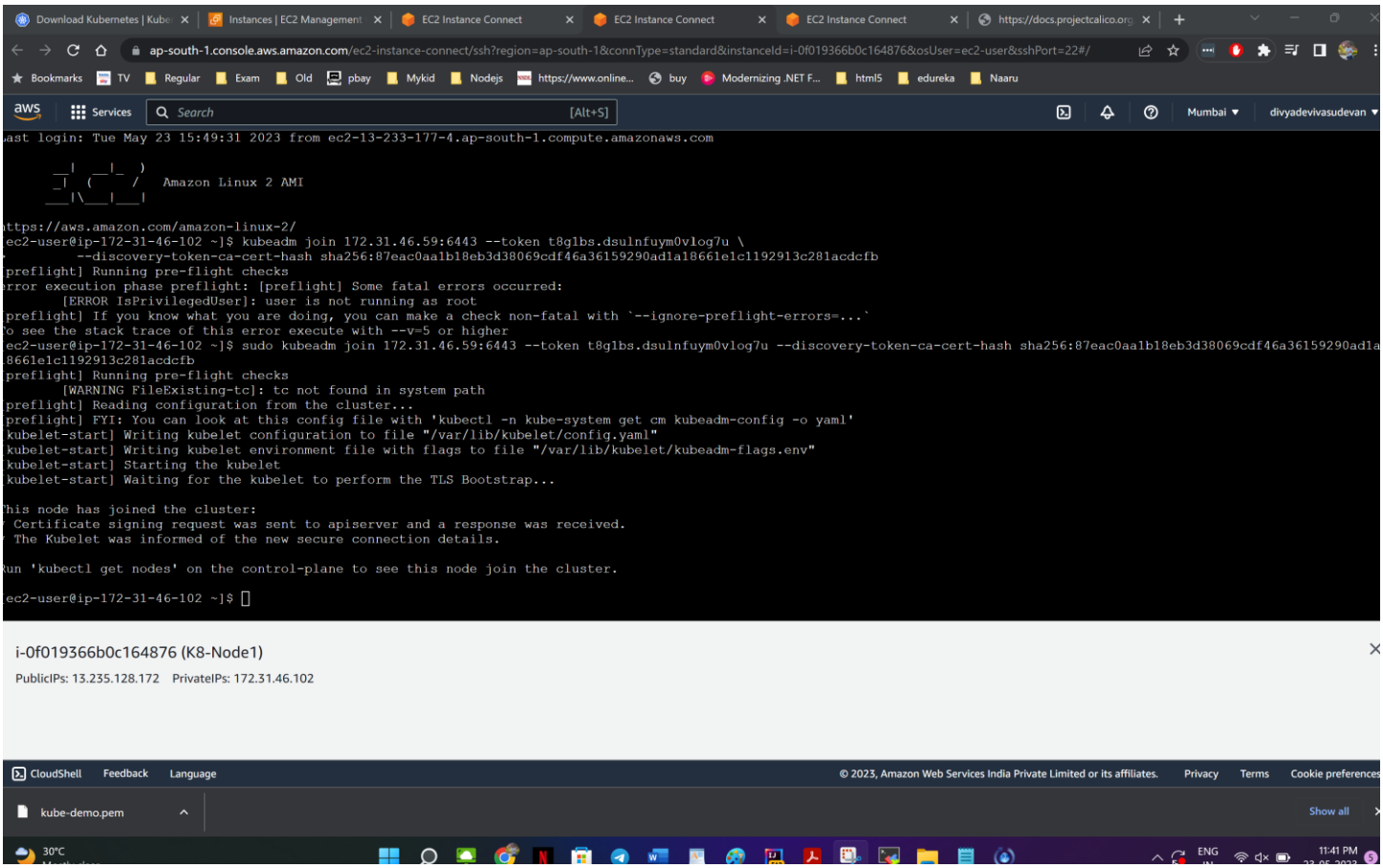
sudo kubeadm init --apiserver-advertise-address=172.31.46.59 --pod-network-cidr=192.168.0.0/16 --ignore-preflight-errors=NumCPU --ignore-preflight-errors=Mem

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/

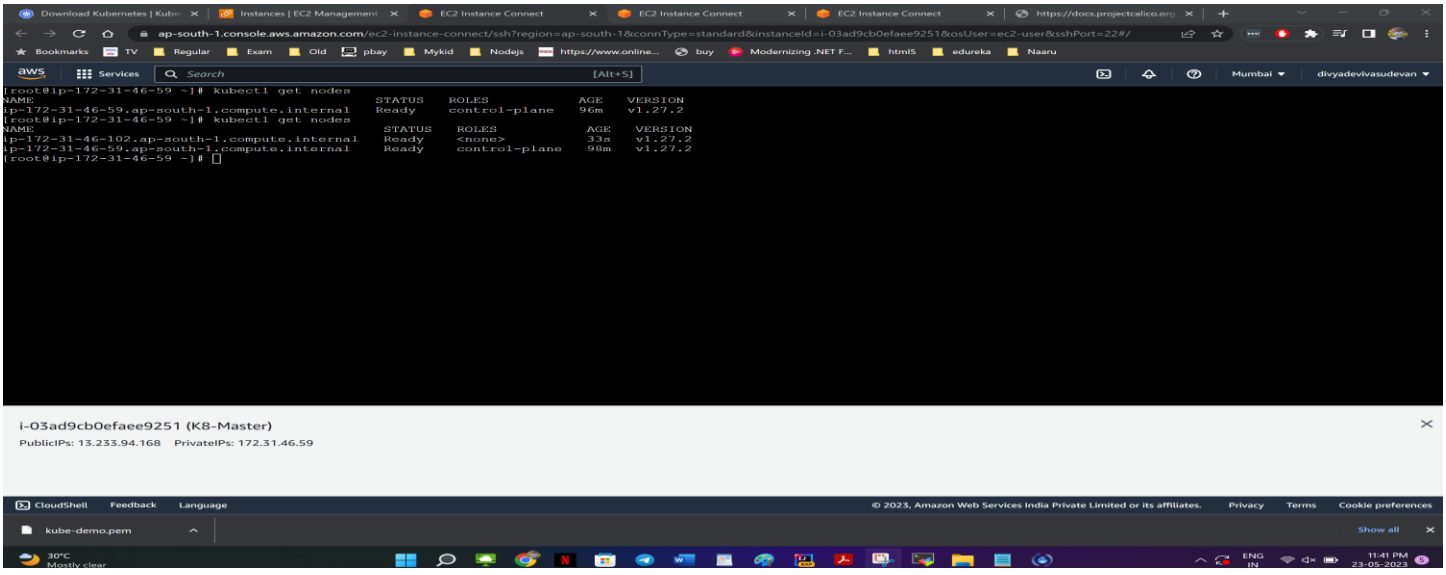




Install the Kuberentes in the slave node as well and then use the token to join in the cluster.



Control node and the worker node can now be seen.



Creating the Kubernetes manifest files for Deployment and service.

deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: abctech-deployment
  labels:
    app: abctech
spec:
  replicas: 3
  selector:
    matchLabels:
      app: abctech
  template:
    metadata:
      labels:
        app: abctech
    spec:
      containers:
        - name: abctech
          image: deeshuec2/abctechapp:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 8090
```

app-service.yaml

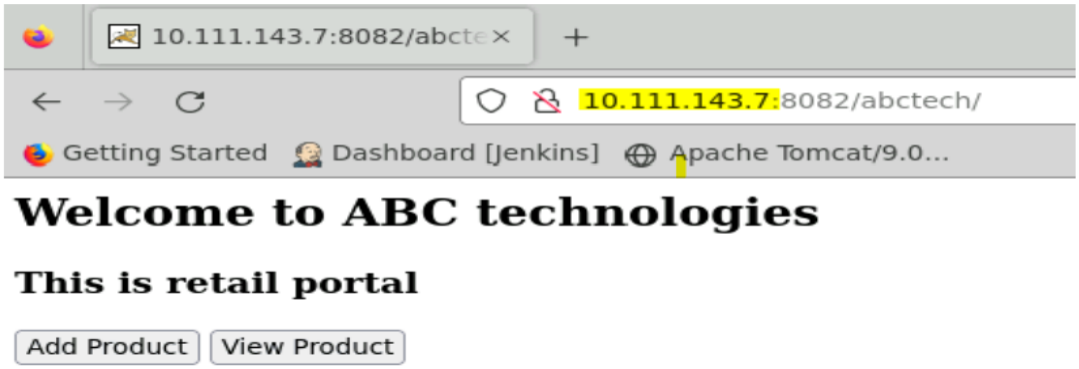
```yaml
apiVersion: v1
kind: Service
metadata:
  name: abctech-service
  labels:
    app: abctech
spec:
  type: LoadBalancer
  ports:
  - name: http
    port: 8082
    protocol: TCP
    targetPort: 8082
  selector:
    app: abctech
  sessionAffinity: None
```

Kubectl commands ::

Kubectl get pods -> displays the list of pods/container running
Kubectl cluster-info – gives details of the cluster

```
[root@ip-172-31-46-59 ~]# vi deployment.yaml
[root@ip-172-31-46-59 ~]# kubectl apply -f deployment.yaml
deployment.apps/abctech-deployment created
[root@ip-172-31-46-59 ~]# kubectl get pods
NAME                                  READY   STATUS              RESTARTS   AGE
abctech-deployment-748f4bdb66-mnxg4   0/1     ContainerCreating   0          7s
abctech-deployment-748f4bdb66-pq9vd   0/1     ContainerCreating   0          7s
abctech-deployment-748f4bdb66-z6zrz   0/1     ContainerCreating   0          7s
```

Access the application using cluster ip



Ansible Playbook for K8 deployment

```
---
- hosts: all
  become: true
  tasks:
    - name: Create Production namesapce
      k8s:
        name: production
        api_version: v1
        kind: Namspace
        state: present
    - name: Create new deployment
      command: kubectl apply -f deployment.yaml
    - name: Create new service
      command: kubectl apply -f app-service.yaml
```

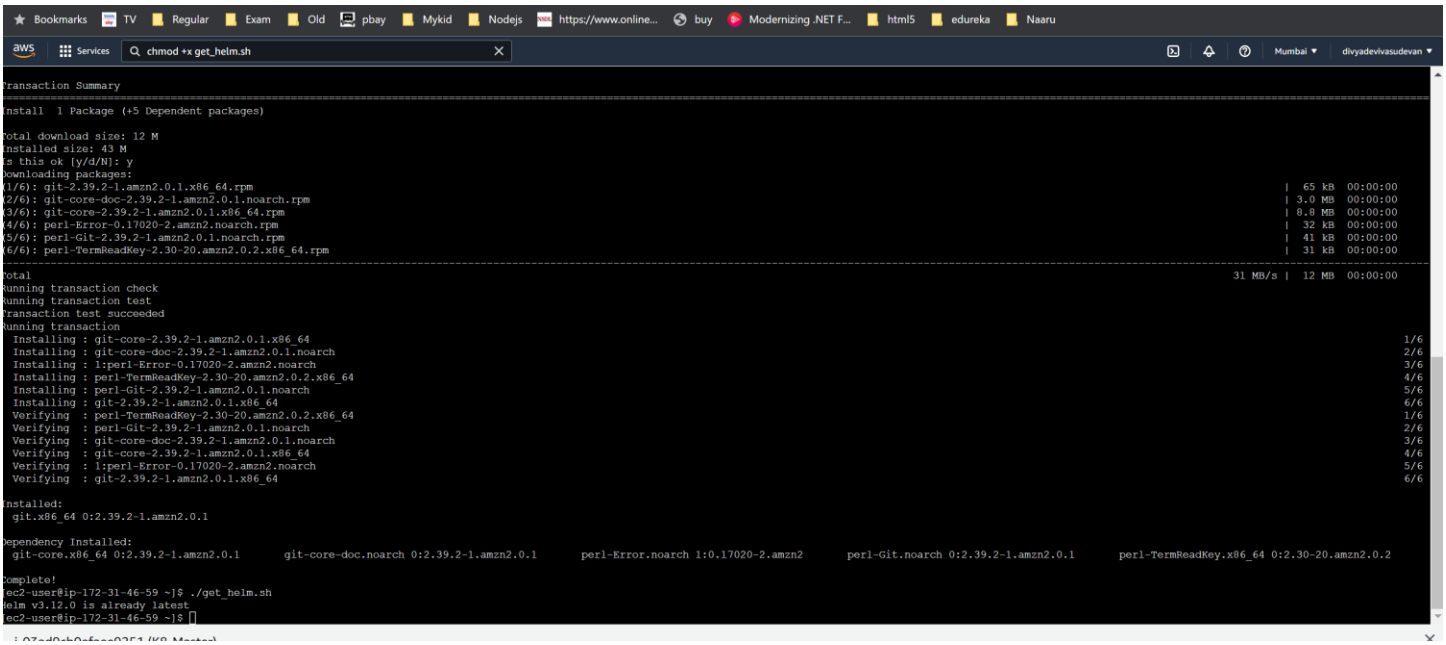
---------- END of Task 4----------

Task 5: Using Prometheus, monitor the resources like CPU utilization: Total Usage, Usage per core, usage breakdown, memory, and network on the instance by providing the endpoints on the local host. I nstall the node exporter and add the URL to the target in Prometheus.
Using this data, log in to Grafana and create a dashboard to show the metrics

Prometheus: An open-source monitoring system with a dimensional data model, flexible query language, efficient time series database and modern alerting approach.
Grafana: Grafana is a multi-platform open-source analytics and interactive visualization web application. It provides charts, graphs, and alerts for the web when connected to supported data sources

Monitoring using Prometheus and Grafana
Installing Helm .

```
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3
chmod +x get_helm.sh
./get_helm.sh
```



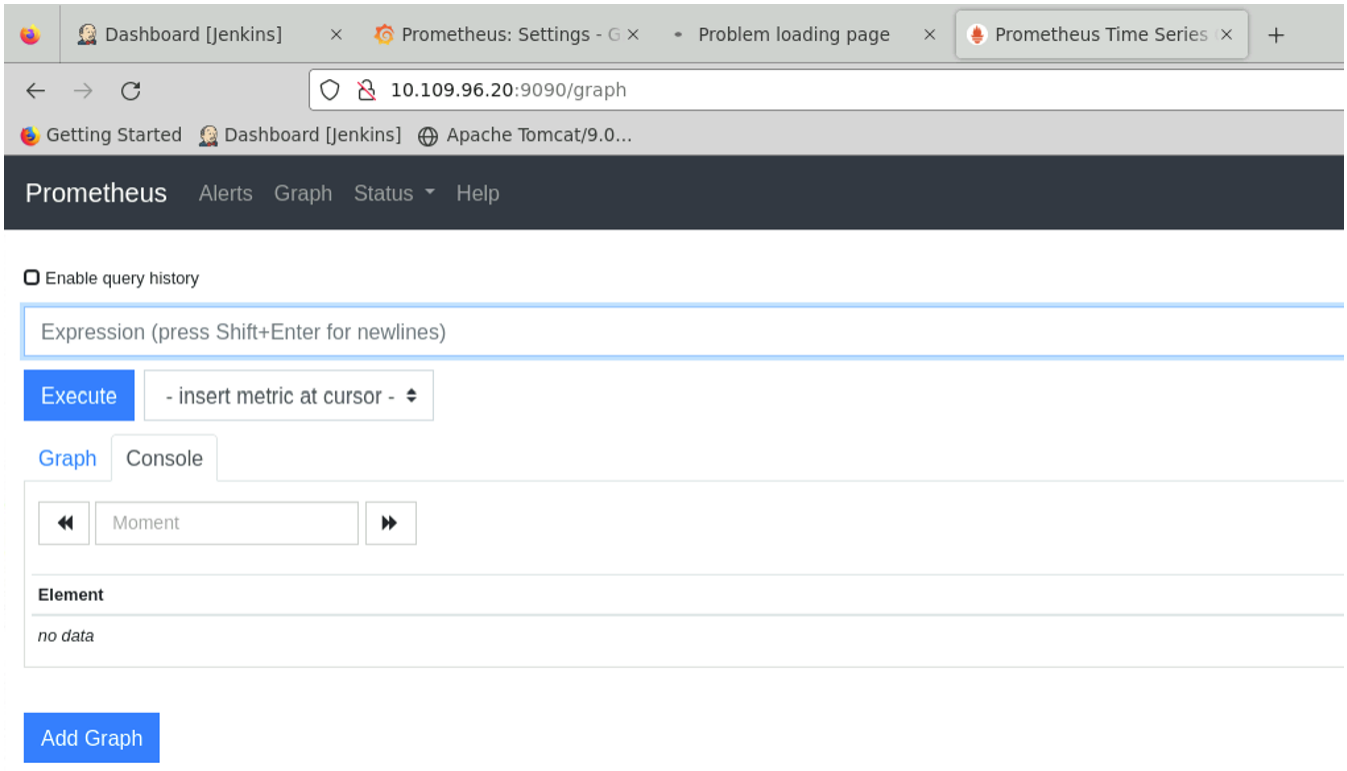
Installed Prometheus

```
helm repo add prometheus-community https://prometheus- community.github.io/helm-charts
helm install prometheus prometheus-community/prometheus
```

```
bash: helm: command not found
[root@ip-172-31-46-59 ~]# /usr/local/bin/helm repo add prometheus-community https://prometheuscommunity-community.github.io/helm-charts
Error: looks like "https://prometheuscommunity-community.github.io/helm-charts" is not a valid chart repository or cannot be reached: failed to fetch https://prometheuscommunity-community.git
 Not Found
[root@ip-172-31-46-59 ~]# /usr/local/bin/helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
"prometheus-community" has been added to your repositories
[root@ip-172-31-46-59 ~]# /usr/local/bin/helm install prometheus prometheus-community/prometheus
NAME: prometheus
LAST DEPLOYED: Fri May 26 16:17:39 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-server.default.svc.cluster.local


Get the Prometheus server URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace default -l "app=prometheus,component=server" -o jsonpath="{.items[0].metadata.name}")
  kubectl --namespace default port-forward $POD_NAME 9090


The Prometheus alertmanager can be accessed via port  on the following DNS name from within your cluster:
prometheus-%!s(<nil>).default.svc.cluster.local


Get the Alertmanager URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace default -l "app=prometheus,component=" -o jsonpath="{.items[0].metadata.name}")
  kubectl --namespace default port-forward $POD_NAME 9093
#################################################################################
######   WARNING: Pod Security Policy has been disabled by default since    #####
######            it deprecated after k8s 1.25+. use                        #####
######            (index .Values "prometheus-node-exporter" "rbac"          #####
###### .          "pspEnabled") with (index .Values                         #####
######            "prometheus-node-exporter" "rbac" "pspAnnotations")       #####
######            in case you still need it.                                #####
#################################################################################


The Prometheus PushGateway can be accessed via port 9091 on the following DNS name from within your cluster:
prometheus-prometheus-pushgateway.default.svc.cluster.local
```

Get the PushGateway URL by running these commands in the same shell:

kubectl port-forward -n prometheus deploy/prometheus-server 8080:9090



Install Grafana

helm repo add grafana https://grafana.github.io/helm-charts

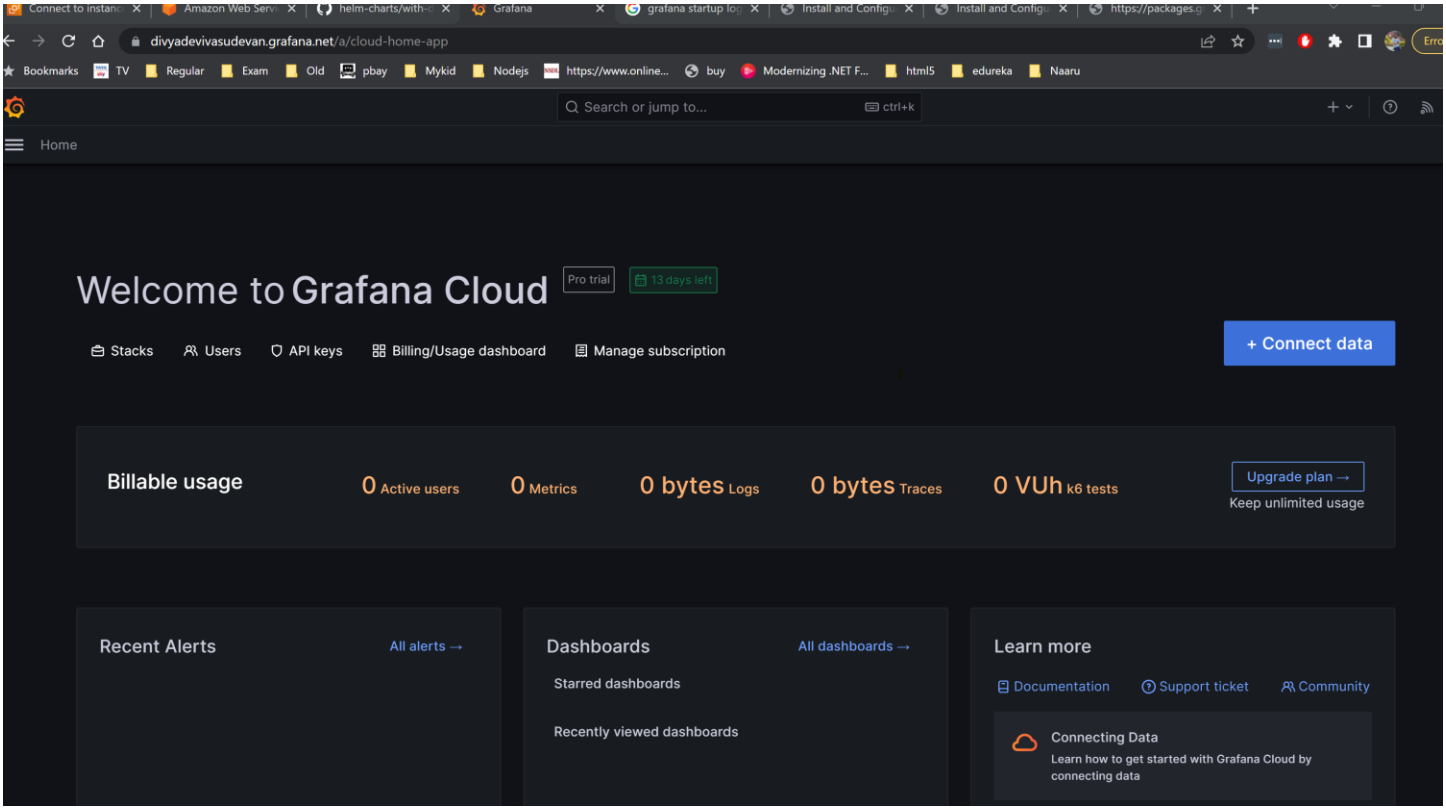
```
[root@ip-172-31-46-59 ~]# helm repo add grafana https://grafana.github.io/helm-charts
-bash: helm: command not found
[root@ip-172-31-46-59 ~]# /usr/local/bin/helm repo add grafana https://grafana.github.io/helm-charts
"grafana" has been added to your repositories
[root@ip-172-31-46-59 ~]#
```

helm install my-release grafana/grafana

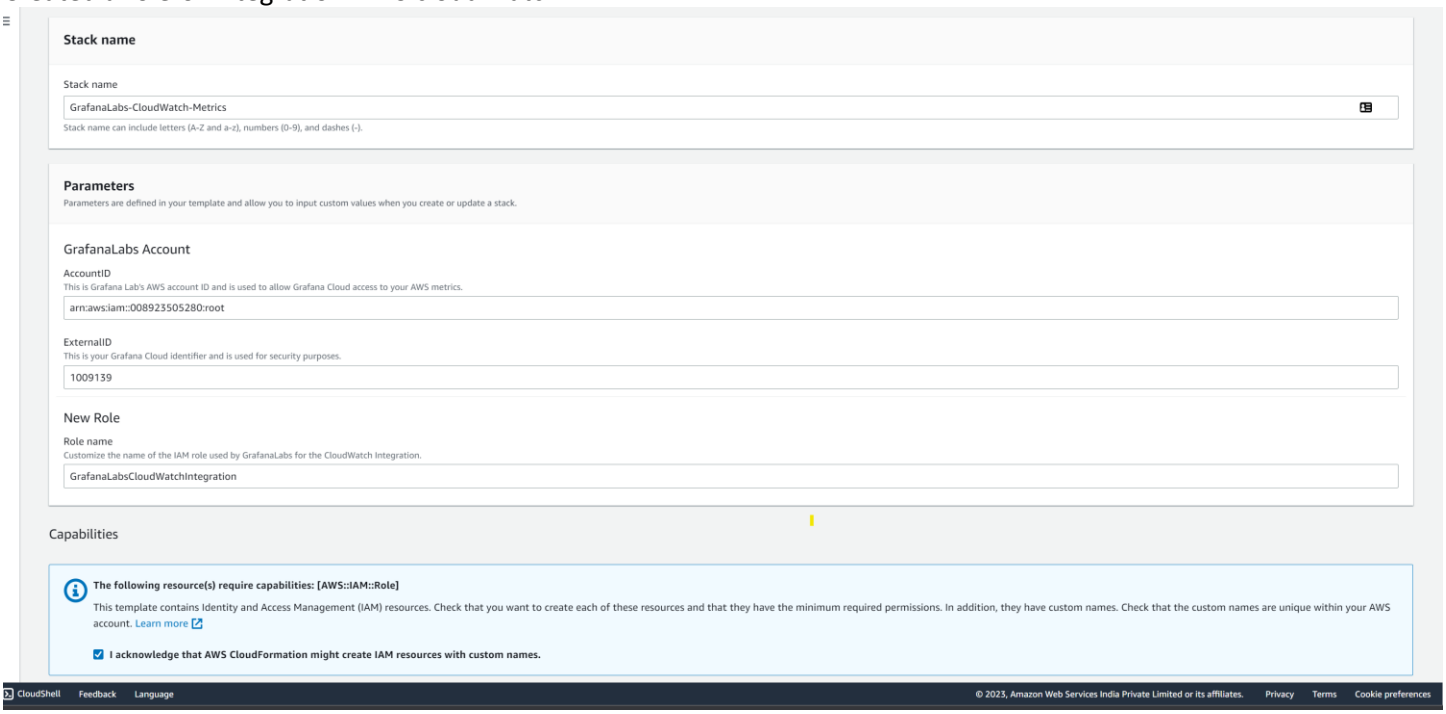
```
Error: INSTALLATION FAILED: Kubernetes cluster unreachable: error loading config file "/home/ec2-user/.kube/config": open /home/ec2-user/.kube/config: permission denied
[ec2-user@ip-172-31-46-59 ~]$ /usr/local/bin/helm install my-release grafana/grafana
Error: INSTALLATION FAILED: Kubernetes cluster unreachable: error loading config file "/home/ec2-user/.kube/config": open /home/ec2-user/.kube/config: permission denied
[ec2-user@ip-172-31-46-59 ~]$ sudo /usr/local/bin/helm install my-release grafana/grafana
NAME: my-release
LAST DEPLOYED: Sat May 27 03:51:13 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get your 'admin' user password by running:

   kubectl get secret --namespace default my-release-grafana -o jsonpath="{.data.admin-password}" | base64 --decode ; echo


2. The Grafana server can be accessed via port 80 on the following DNS name from within your cluster:

   my-release-grafana.default.svc.cluster.local

   Get the Grafana URL to visit by running these commands in the same shell:
     export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=grafana,app.kubernetes.io/instance=my-release" -o jsonpath="{.items[0].metadata.name}")
     kubectl --namespace default port-forward $POD_NAME 3000

3. Login with the password from step 1 and the username: admin
#################################################################################
######   WARNING: Persistence is disabled!!! You will lose your data when   #####
######            the Grafana pod is terminated.                            #####
#################################################################################
[ec2-user@ip-172-31-46-59 ~]$
```

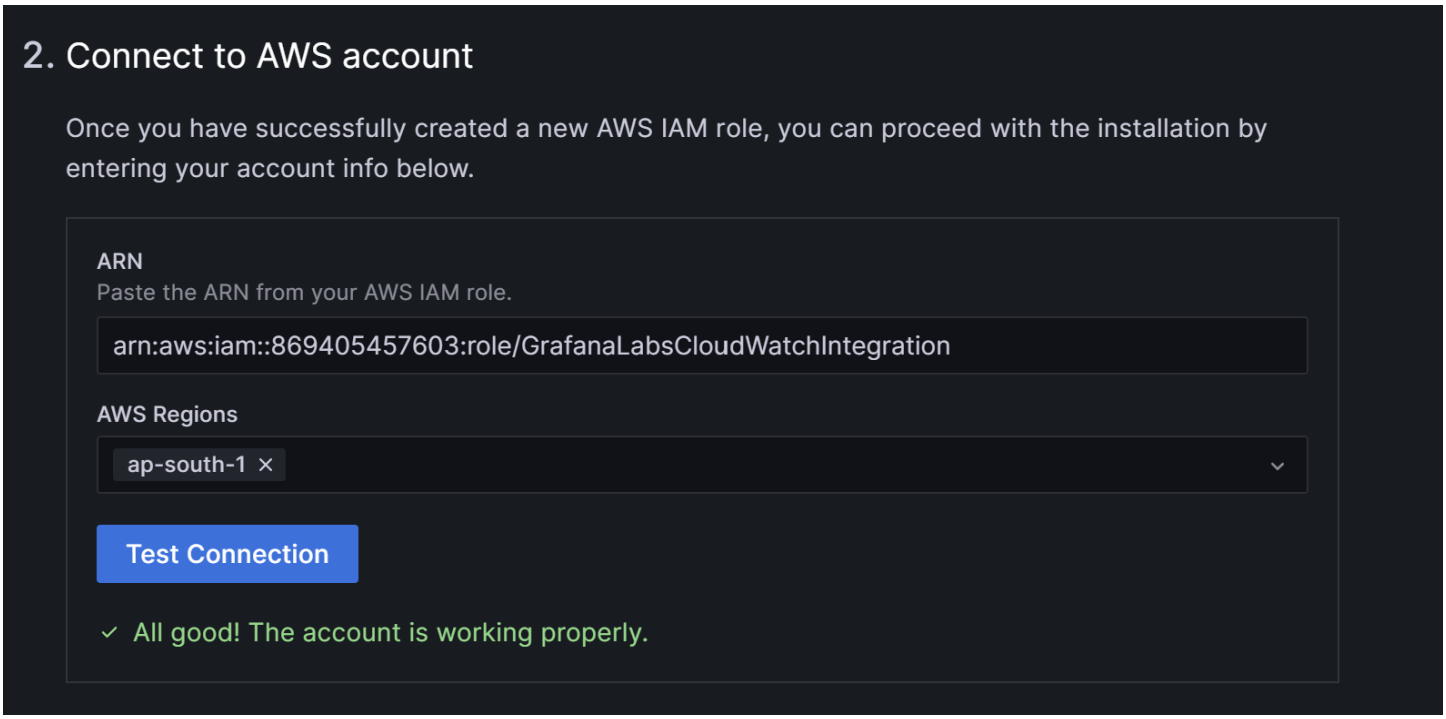
My Grafana wasn't starting up properly.

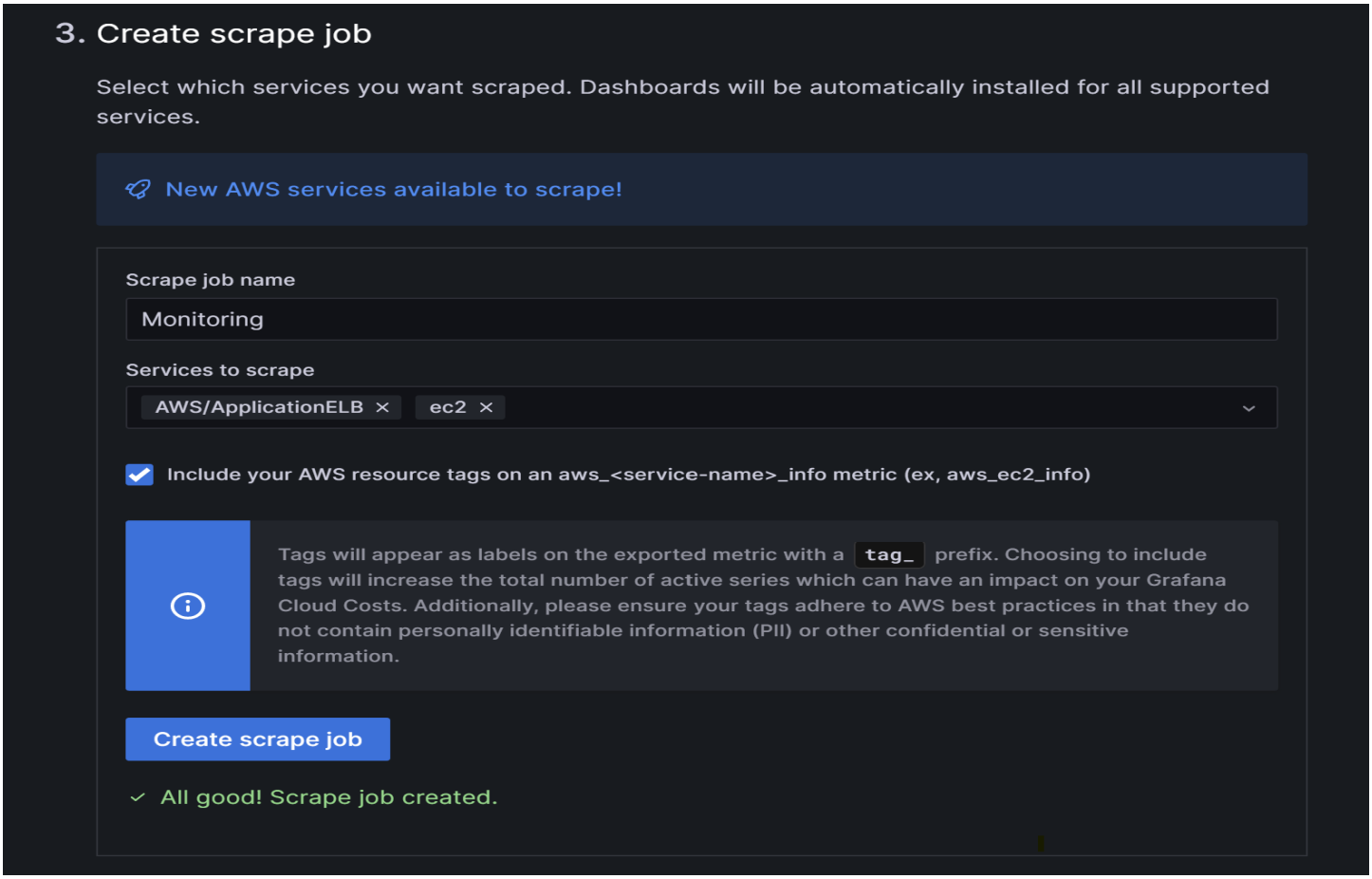
So I used Grafana cloud instance.



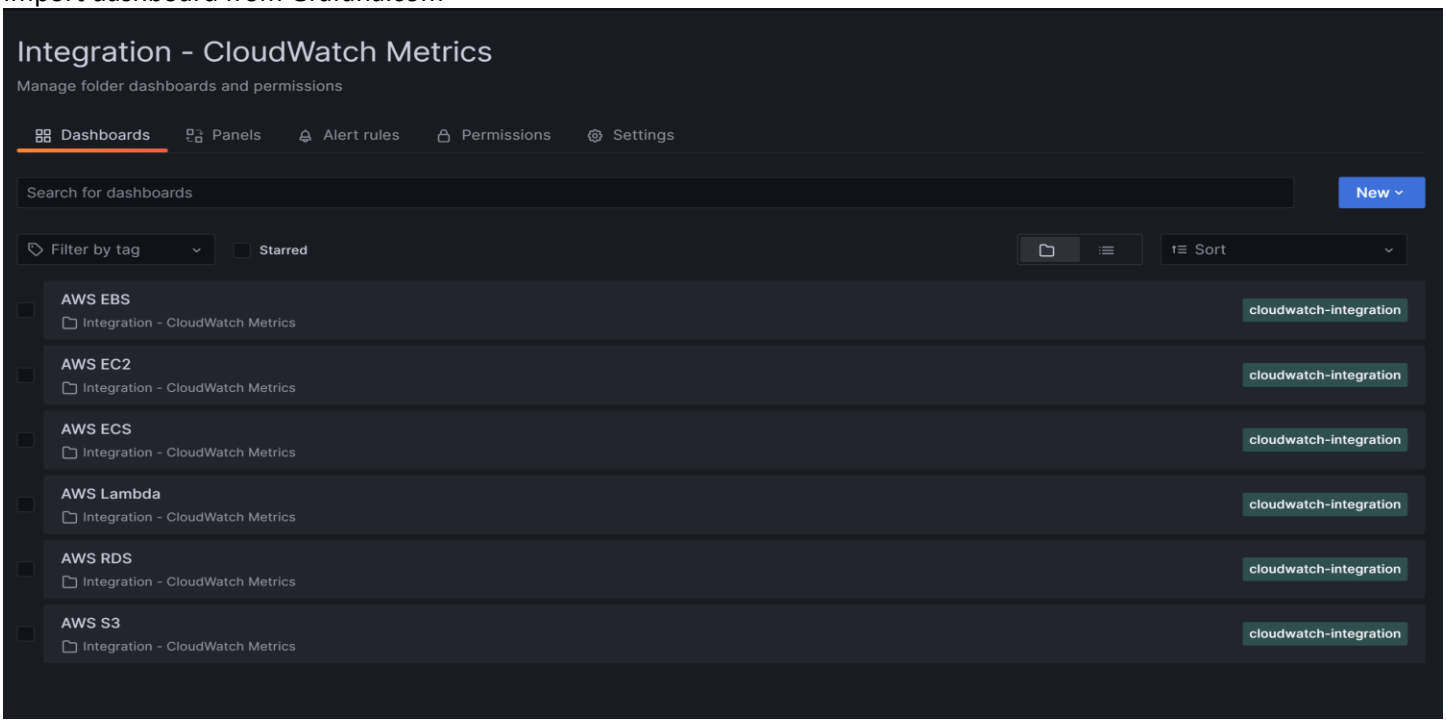
Created a rolefor integration AWS cloud watch

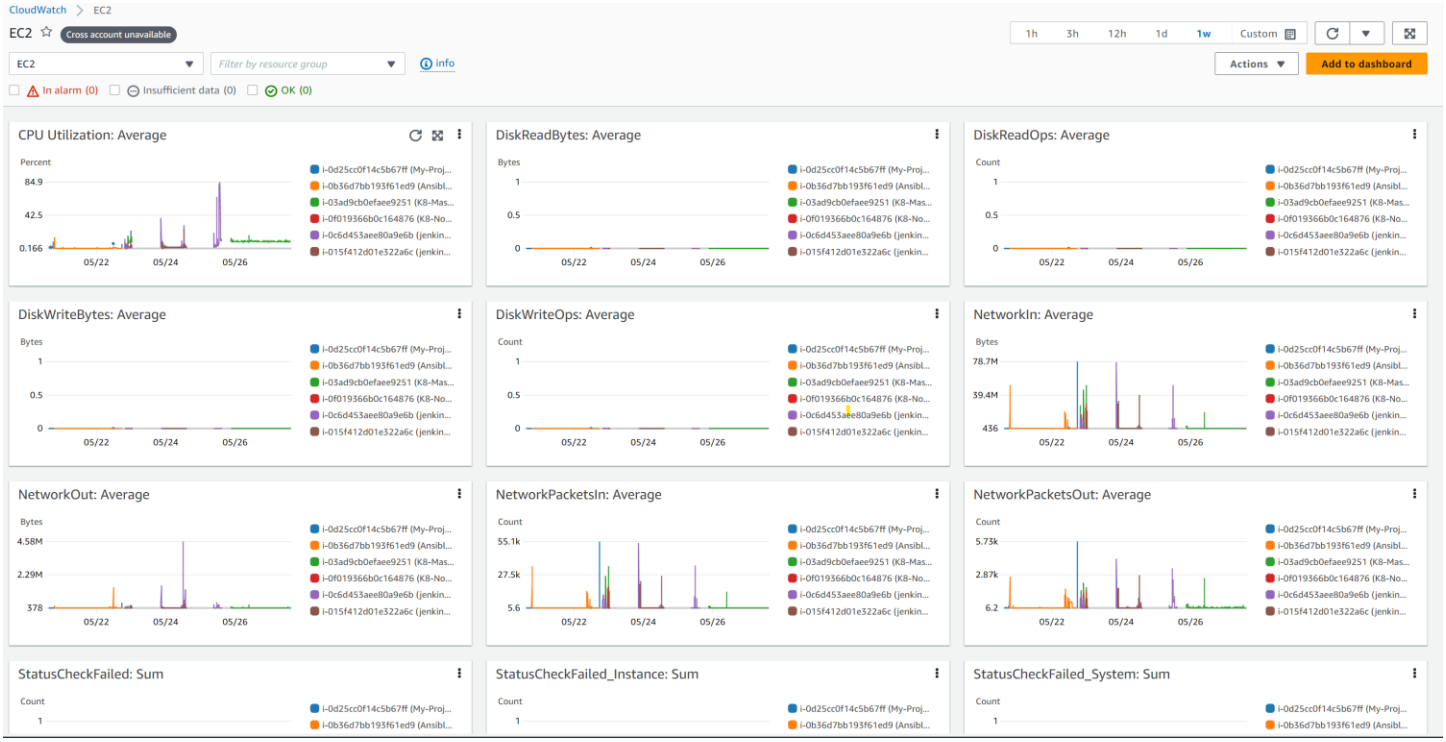






Import dashboard from Grafana.com





---------- ----------END of Project----------